RELIABILITY STUDY OF FLASH MEMORY AND ITS APPLICATIONS

A Thesis

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirement for the Degree of

Master of Science

By

SHUO WU

August 2011

ABSTRACT:

Flash memory cells are typically quantified as extremely reliable as their endurance cycles have far exceeded the requirements of the actual application. However, as the dimension of device shrinks further in the near future, the reliability issue in flash memory cells has become a major concern in industries and evokes many research interests in universities worldwide. This thesis proposes a reliability model based on three contributing factors: stress-induced leakage current, random telegraph noise and reading errors caused in peripheral circuits. Each of the three factors is modeled mathematically as a function of threshold voltage. The resulting model is a probabilistic one where it indicates the probability of an error read at a given time. Numerical variable values used in the probabilistic models are extracted using Simulated Annealing algorithm. Simulation results are run in MATLAB environments and are compared to experimental data to validate the accuracy and consistency of the model. In addition, a random number generator (RNG) using the randomness factors in RTN phenomenal is devised. The detailed generating algorithm and feedback loop scheme are introduced. The RNG is then tested using exploratory data analysis techniques, theoretical statistical tests and an industry standard NIST test suites.

BIOGRAPHICAL SKETECH:

Shuo (Steven) Wu was born on December 10, 1987 in Beijing, China. He was raised in a loving family and enjoyed his childhood in suburban Beijing. He immigrated to Toronto, Canada with his family at the age of 17. As a teenager, he excelled in mathematics and sciences and received numerous awards in provincial and national mathematical contests.

He attended the Engineering Science Program at the University of Toronto in 2005. He received the *University of Toronto Scholar award* and the *Queen Elizabeth Scholarship* for four consecutive years. During his undergraduate study, he served as a Research Assistant in the Intelligent Transportation Lab and the Wireless Multi-User Research Lab at the University of Toronto. He worked as a design verification engineer at *Advanced Micro Devices Inc.* Markham Campus as part of the Professional Engineering Year program from May 2009 to August 2010. His undergraduate thesis was "*Complicated Simulation in Wireless Communication networks using CUDA parallel computing*" under the supervision of Professor T.J. Lim. He received Bachelor of Applied Science Degree with honors in Engineering Science (Major in Electrical Engineering) from University of Toronto in May 2010.

Shuo Wu attended Cornell University since August 2010 and received the Cornell University Fellowship. He has been working with Professor Edwin C. Kan and Professor G. Edward Suh on the topic of reliability study in flash memories in pursuit of his Master of Science Degree.

To My Parents

ACKNOWLEDGEMENT

I feel very fortunate to have learned from and have worked with so many incredible individuals at Cornell University during my master study. Foremost, I thank my advisor Professor Edwin C. Kan for his guidance and support throughout the past year. I have benefited tremendously from his unique blend of energy, vision, technical insights and practical sensibility. I thank my associate advisor Professor G. Edward Suh for being a great teacher and a friend. I will miss the fun and intellectually stimulating environment in the weekly group meeting.

I also thank many of my colleagues with whom I have had the good fortune to collaborate. Joint work with Ph.D. student Yinglei Wang led to the main results of this thesis. I have benefited from many hours of stimulating discussion with her. I also owe her a great deal for her friendship. In addition, I thank K.K. Yu and Greg Malysa from Trustworthy Systems Group at Cornell University, for whom that I had opportunities to collaborate on various subjects. Finally, I express gratitude to my deeply beloved girlfriend Kun Hu for inspiring and accompanying me throughout my graduate school studies.

I acknowledge the Cornell University Fellowship program for providing funding for my Master's study and support.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

1. Introduction

1.1 Background

Flash memories are non-volatile storage chips that can be easily programmed and erased by end users. In the past two decades, the tremendous advancement of design and manufacturing technology in semiconductor industry lead to a drastic transformation in personal computers and mobile devices. As the Moore's Law continues as of today, the number of transistors on a single die has doubled almost every two years [1]. This is certainly no exception with flash memory chips. With the state of art technology in 2011, the size of transistors of flash memory could be as low as 20 to 25 nm, which leads to a density as much as 230 Gigabytes per inch square for multiple level cell flash chips shown in an industry demonstration a couple years back.

The advancement of flash memory technology has played a significant role in information technology in the last decade. The costs of the flash memories are far less than byte-programmable EEPROM, or electrically erasable programmable read-only memories. Therefore, it has become the dominant technology wherever a significant amount of non-volatile and solid state drive is needed. The flash memories were originally mainly used in portable storage devices, digital camera memory cards or digital audio players. Recently, the popularity in smart phones and tablet computers lead to a shift in flash memory market shares. According to a latest study by Gartner, more than 50% of flash memories produced are used in these mobile devices in 2011. The flash memory industry is dominated by large players such as Samsung, Toshiba and Micron

Technology, where they combine and produce annual revenue in excess of 20 billion US dollars.

Although flash memory is a groundbreaking technology, it was invented not long ago by Japanese engineer named Shoji Ariizumi working for Toshiba in 1980. The flash memory works as a floating-gate transistor where it stores the information in the memory cell. The traditional single-level cells (SLC) devices store one bit per transistor, whereas the newly invented multi-level cell, or MLC devices, can store multiple bits per transistor with different applied electrical charges. There are two main types of flash memory, the NAND type and the NOR type. The connections of the individual cells and the interface between reading and writing are different for these two types. However, the NAND type has been the dominant technology due to its lower cost and higher density.

In the past thirty years, many researchers in universities and technology corporations have been working to understand the physics behind flash memories, thus to improve the transfer rate, density, reliability and the overall performances of flash memory cells. This imposes a unique but challenging problem to researchers and engineers worldwide. This thesis will present the proposed theoretical model based on physical behaviors, experiments setups and analyses performed in order to understand the reliability issue of flash memories on a system level.

1.2 Motivation

Historically speaking, flash memories are quantified as extremely reliable and their endurance cycles have far exceeded the requirements of the actual application. For example, in a typical flash memory application such as portable disk drives or cellular phones, the memory information is programmed and erased no more than a thousand times in its life cycle. However, modern flash memory chips from major manufactures have a mean time to failure time on the order of 100,000 endurance cycles.

The extreme reliability of flash memory chips leads to the ignorance of reliability study in industries. However, as the transistors size gets reduced further and multiple-level cells in flash memories are used, the behavior of flash memory in stress have recently attracted many attentions in both academia and industries. Flash memory chips pose an interesting but challenging problem in terms of reliability compared to other integrated circuits. They are composed of the same basic materials and processing steps as other types of integrated circuits and similar physical degradation process apply, such as oxide breakdown, hot-electron damage, failure-in-time rates and bathtub curves. However, flash memory chips are nevertheless significantly different from other types of chips, thus lead to unique and challenging reliability characteristics.

Flash memories are designed to be able to program, erase and retain charge on its floating gates. The uniqueness of this functionality requires higher electric fields than other traditional devices due to its reliance on oxide tunneling and hot-electron injection for its operations. Reliability issues with these mechanisms have somewhat impede the scaling issues of flash memory chips. In addition, the functionality of flash memory relies

on analog operations within the chip. In another words, the digital information of bits are stored in analog forms as threshold voltage distribution, then the analog information is later read and translated into digital information. In fact, the control circuits are analog while all other circuits, including charge pumps, sense amplifiers and voltage regulating circuits are all analog. With the state of art technology in multiple level cell flash memories, multiple bits of information are stored in one physical cell by different but precise placement and readout of threshold voltages. Obviously, the precise control of these analog threshold voltages would have major impact on the reliability of flash memory chips.

It is essential to state that there are two major modes of operations for flash memory cells: low-voltage operations (the powered down mode) and high voltage mode (the operation of program and erase). The low voltage operations are similar to other traditional integrated circuits while the high voltage mode is unique. Typically, the flash memory cells fail due to the high voltage operations. This thesis study will focus on the behaviors of cells after many program and erase cycles and thus learn the reliability behaviors of flash memory cells.

In the past, there have been some sporadic studies relating to the cause of failure of flash memories. Researchers speculate the failures might be related to cycling-induced degradations in flash memories, stress-induced tunnel oxide leakage current, process-impacts on flash memory reliability and high-voltage periphery circuit variation. These above phenomenal might all contribute to the reliability problem of flash memory. However, as the transistor size gets reduced significantly and approaching the diameter of

an atom, the impacts of random telegraph noise, or commonly known as RTN, could be the biggest factor affecting the reliability of flash memory in the future.

Random Telegraph Noises have been a major concern for analog and radio frequency in CMOS technologies. It is also projected to be a serious issue in flash memory chips as the dimension of transistors shrink. The voltage threshold caused by random telegraph noise could be estimated by the formula below:

$$V_{th} = \frac{q}{L_{eff} \times W_{eff} \times R_c \times C_{ox}} \quad (1)$$

where q is the elementary charge, L_{eff} is the effective channel length, W_{eff} is the effective channel width, R_c is the coupling coefficient of control gate to floating gate and C_{ox} is the capacitance of the gate dielectric. The coupling coefficient amplifies the threshold voltage due to the stack-gate structure in flash memories. In addition, it is difficult to enlarge C_{ox} to reduce oxide thickness due to leakage current. As a result, the reduction in dimension of transistors would lead to an abrupt increase in threshold voltage where it negative affects the performance and reliability of memory chips.

Beside random telegraph noises, stress-induced leakage currents and variation in peripheral circuits are the two other main reasons lead to reliability problems. It is advantageous to reduce the tunnel oxide thickness thus increases scalabilities. However, this leads to increase in leakage current and it causes serious issue in data retention and read disturbance. In addition, reading errors in peripheral circuits would also be a major reason that causes failures in flash memory cells. Although both of the behaviors were well studied in the past years, researchers rarely considered the incremental effect of

random telegraph noises, thus not accurately model the behavior of flash memory cells in stress.

The motivation of this thesis is to build a generic model considering the effects of random telegraph noises, stress-induced leakage current and peripheral circuit reading errors. It is intended to come up with a model involving the three major factors above to explain the failure behaviors of common flash memory chips. With the proposed model, its main objective is to better understand flash memories in stress and the main reasons that lead to breakdown. After fully understanding the breakdown behavior of flash memory chips, there are many interesting applications that can be applied with the knowledge. First of all, the random telegraph noise is a stochastic process and this characteristic could help devise a random number generator using common flash memory chips. In addition, different bits could be characterized by leakage current, RTN and peripheral circuit parameters and this information could essentially be identified as signatures of certain flash memory chips. Last but not least, if the failure behavior is entirely characterized and fully understood by this model, better error correction codes could be developed in order to increase the mean-to-failure time for flash memory chips.

1.3 Overview

The thesis is divided up into three main components. First of all, a literature review is conducted to study the researches done by people at other universities or industries on the topic of flash memory reliability. This includes the current knowledge about stress-induced leakage current, random telegraph noises, reading errors in peripheral circuits and random number generation techniques using physical mechanisms. Secondly, the proposed model to explain the failure behaviors is presented in details. The physical meanings of the six parameters used in the model are explained as well. The experimental procedures performed on flash memory chips and the results are documented. Signal processing techniques are applied on raw experimental data in order to analyze them in a systematic way. MATLAB simulations are also performed to verify the proposal model, and the simulation results are compared with actual results received from experiments. Last but not least, the applications using the proposal model are illustrated in the third part of the thesis. It mainly presents the random number generation mechanism applying the random nature of RTN behavior. It shows the signal processing technique used as well as the loop feedback algorithm needed to detect the suitability of various bits. The performance of this random number generator is compared with other random number generator schemes. The conclusion of the thesis summarizes the main results achieved from the study and explains the future direction of researched needed to better understand the reliability issues in flash memory cells.

2. Literature Review

The literature review section is divided up into four components. The first part writes about the current research in the area of stress-induced leakage current. The second component illustrates the research performed on random telegraph noises and the main characteristics of RTN. The third section explains the current knowledge on peripheral circuits and the reading error associated with it. The three factors above all contribute to the overall failures of flash memory cells thus are significant to flash memory reliability studies. The last part presents the current schemes used in random number generator and the benchmark criteria associated with physical model based random number generators.

2.1 Stress-Induced Leakage Current

As the dimension of transistor shrinks, reducing the thickness of tunnel oxide is a technique used to reduce the voltage used in programming and erase operations. Flash memory chips use high-field injection of electrons through a layer of very thin tunnel oxide to charge or discharge the floating gates in order to perform programming and erase operations. It was studied as early as 1985 by D.A. Baglee and M. Smayling, in their paper "The effects of Write/Erase cycling on Data loss in EEPROMs" [2], they claimed that high-field stressing of the gate oxides increases the low-field leakage current and named it as stress-induced leakage current, or in short SILC. It was also studied by scientists lead by F. Masuoka in 1993[3] where they concluded that SILC could cause serious issues on data retention and read disturbance in flash memory chips and it becomes more severe with reduction of transistor size. In S. Ahn's paper "Scaling Down

of Tunnel Oxynitride in NAND Flash Memory"[4], they modeled SILC as a trap-assisted tunneling process and caused by high stressing and generation of oxide electron traps. Indeed, the stress-induced leakage current depends on many factors, including but not limited to stress field, waveform, polarity, oxide thickness, time, and oxide fabrication variations.

Although stress-induced leakage current is a challenging subject, it still generates intensive studies on the topic. P. Kuhn and others, in their paper "Statistical Modeling of the Program/Erase Cycling Acceleration of Low Temperature Data Retention in Floating-Gate Nonvolatile Memories" published in 2002 [5], established a statistically accelerated reliability model and it tried to generate reliability failure rate that incorporated many determining factors including retention time, cycle count, temperature, cycling voltage and other conditions. It was concluded that the statistical distribution of the cells were well behave despite the difficulty to model many individual characteristics such as occasional erratic erase current. With this model, it has been demonstrated that stress induced leakage current related tail of statistical distributions have an extreme-value shape and the threshold voltage shifts with logarithm of time. This is consistent with exponential or quasi-exponential voltage characteristics for leakage current.

It was also shown that the difference in manufacturer technology could also lead to various SILC-related retention loss or reliability issues. In the paper "New Reliability Model for Post-cycling Charge retention of Flash Memories" published by N. Mielke in 2002 [6], it showed that erasing operations were the major causes of damage, indicating that hot-hole injection being an important source of damage. For another manufacturer, it was demonstrated that leakage current was mainly caused by drain-disturb-induced hole

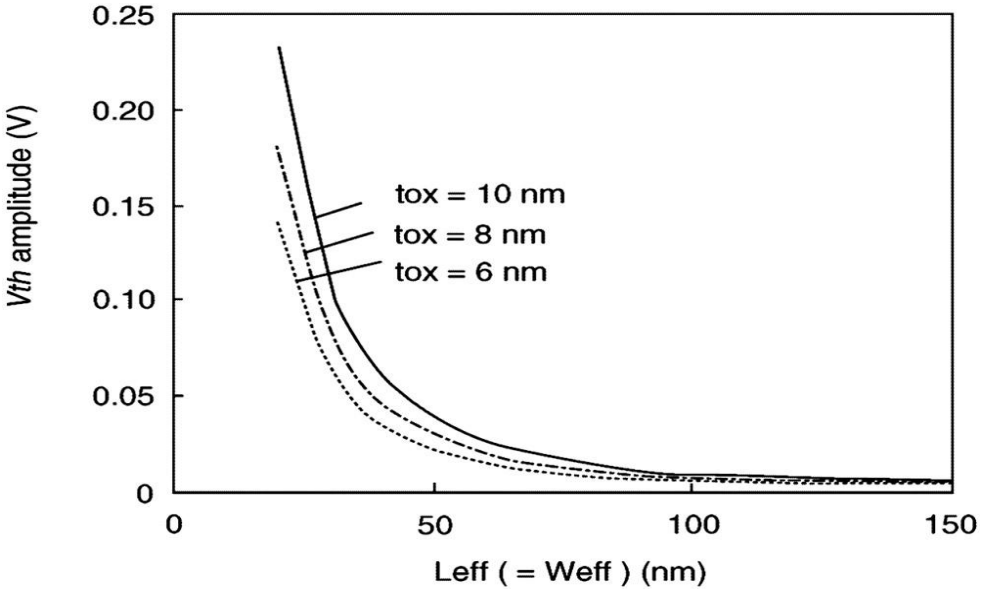
injection and the experiment results exposed that cells with the most drain disturb exposure had more leakage current [7]. Lastly, in Kuhn's paper published in 2002[8], it was shown that 99% of the defection due to leakage currents were caused by the source, or in other words, caused by erase operations. These above papers obviously showed contradictory results and it indicated that vast variation involved in band-to-band tunneling in existences of hot hole in various manufacturing technologies. This imposes the difficulty to have a complete and comprehensive model that explains the reliability behavior of many flash memory cells.

In addition, quantum-mechanical models for trap-assisted tunneling mechanism were developed to explain the behavior of individual cells caused by leakage currents in G. Tempel's paper published in 2004 [9]. It was agreed in scientific community that stress induced leakage current lead to data retention loss in flash memory cells and trap-assisted tunneling consist of two traps in the most common case. In recent researches, scientists moved away from one-dimensional WKB-based models to comprehend three-dimensional effects and enhancement from phonon-assisted tunneling. As a result, non-exponential characteristics of the leakage current were observed for individual flash memory cells [10]. However, for the simplicity of this proposed model, stressed induced leakage current is assumed to behave as an exponential distribution throughout the analysis. The detailed assumption and proposed model is explained in details in Section 3.

2.2 Random Telegraph Noises

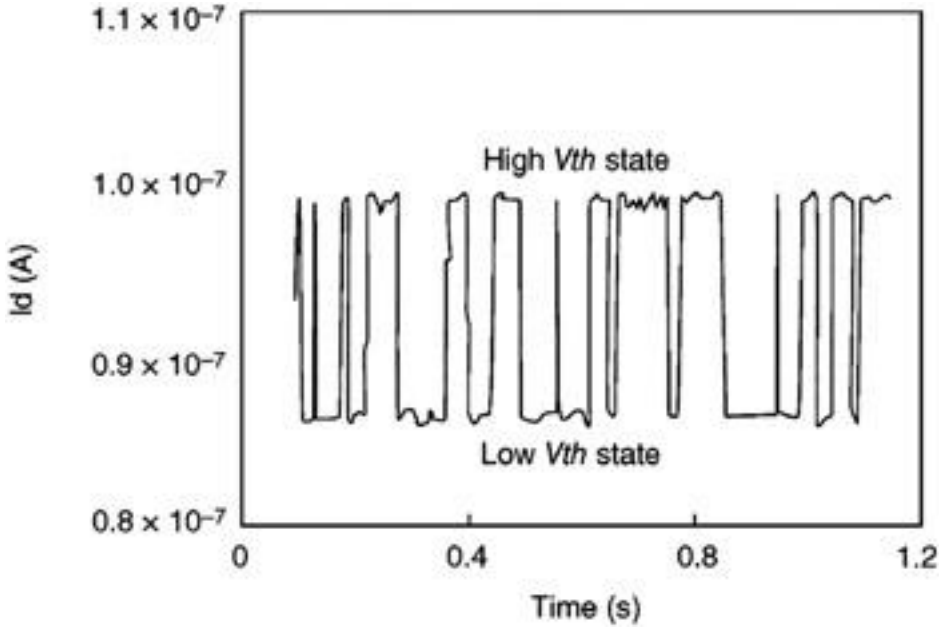
Random Telegraph Noises, or commonly known as RTN, are a major concern in analog and radiofrequency application in CMOS technology. It consists of a sudden step-like transition between multiple discrete voltage levels at random and unpredictable times. Each shift in offset voltage lasts from several milliseconds to a few seconds. As the dimension of transistor shrinks, RTN will become a serious issue in flash memory cells. It was explained in Section 1.2 that reduction of transistor size by half would lead to a four time increase in the voltage threshold of one trap RTN. In the paper "Multi-Level NAND Flash Memory with 63nm-Node TANOS (SiOxide-SiN-Al_2O3-TaN) Cell Structure" published by K. Kim and others in 2006 [11], they believed that random telegraph noises will become a major bottleneck in performance for multiple level cell flash memories. The stack-gate structure increases the threshold voltage multiple times by the coupling coefficient. In addition, the oxide capacitance cannot be increased due to the fact that it is difficult to reduce gate oxide thickness because of leakage current.

Figure 1: Effective channel length vs. Threshold voltage amplitude



As shown in **Figure 1** on previous page, the amplitude of threshold voltage caused by random telegraph noises increase significantly as the dimension of device shrinks. For example, the threshold voltage could be in excess of 100 mV at the 45 nm node. With the state of art technology in the 22nm range, the threshold voltage jump caused by RTN could be well over 200 mV or more. In paper "The Impact of Random Telegraph Signals on the Scaling of Multilevel Flash Memories," published by O. Tsuchiya and others in 2007 [12], they used experimental results to demonstrate that random telegraph noise have become a significant concern in design and manufacturing of multiple level cell flash memory operations. They also illustrated that variations in threshold voltage is caused by random telegraph noise and confirmed the existence of tail bits generated by random telegraph noise, as illustrated in **Figure 2** below.

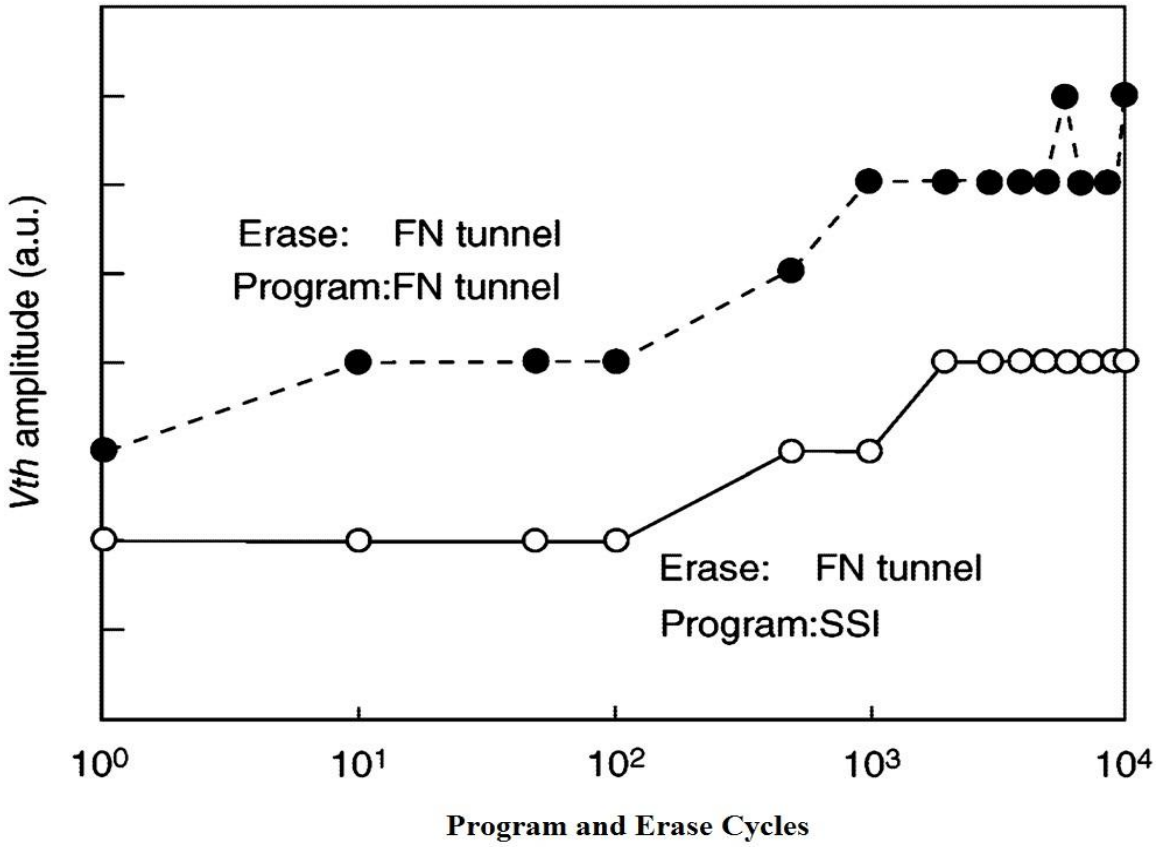
Figure 2: Example of time-series change in drain current in flash memory



In addition, it was shown that additional interface traps would be generated after many program and erase (P/E) cycles. In O. Tsuchiya's paper "The Impact of Random Telegraph Signals on the Scaling of Multilevel Flash Memories" [13] published in 2006,

he reported experimental results where increased P/E cycles led to increased threshold voltage caused by RTN effects. It is shown in **Figure 3** below that the amplitude of threshold voltages more than doubled after ten thousand program and erase cycles.

Figure 3: Amplitude of threshold voltage caused by RTN vs. P/E Cycles



As the dimension of device shrinks further in flash memory cells, the number of electron representing one bit reduced further in multi-level cell flash memories. For example, it is shown that two states are differentiated by less than 100 electrons in a sub-40 nm floating gate MLC devices. Currently, there has little knowledge on the full impact of reliability caused by RTN in flash memories and it would definitely be an interesting area of research in the near future.

2.3 Peripheral Circuits caused Reading Errors

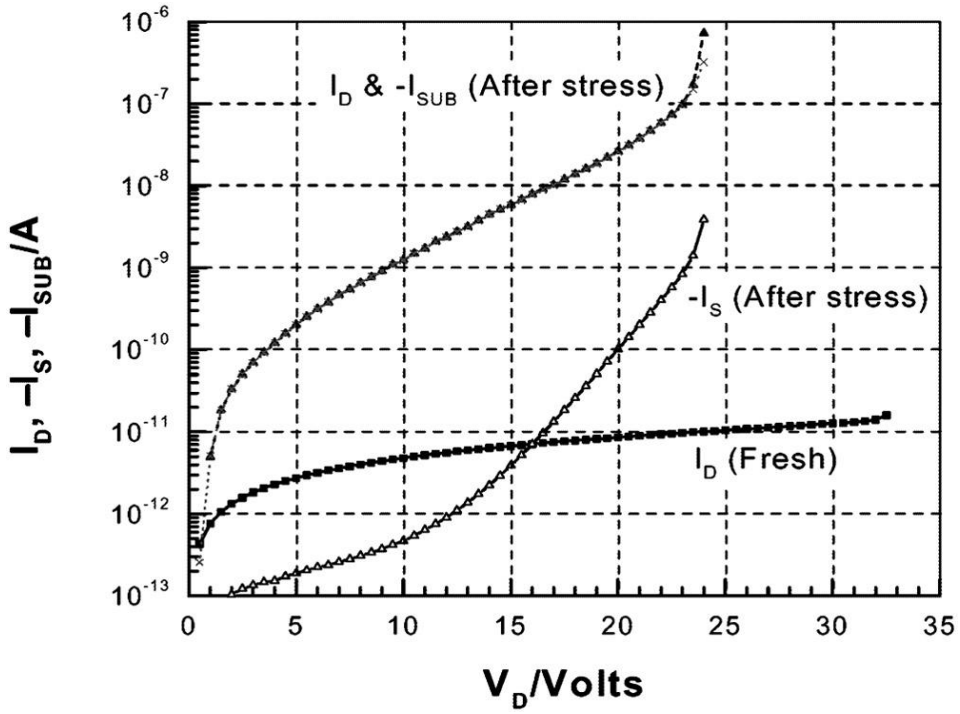
As stated previously in Section 1.2, flash memory cells require high voltage to complete the operation of program and erase. Typically speaking, the voltage could reach as high as 25V for NAND and poly-to-poly tunnel-erase flash memories. The tunnel oxide is not expected to get reduced further due to concern related to stress-induced leakage current. As a result, the voltages required for tunneling program and erase cycles are not expected to scale down further.

Circuits that control the memory array, commonly known as peripheral circuits, use low voltage for read operations and high voltage for program and erase operations. Low voltage needed could be as low as 3.3 V while the high voltages might reach up to 25 V. The vast difference in voltage requirement leads to the usage of dual-gate oxide technology, where one type provides high voltage with thicker oxide while the other provides high-performance low-voltage transistors with thinner oxide. The exact voltage and oxide thickness depend on the manufacturing technology as well as the types of flash memory cells.

The reliability of peripheral circuits, mainly in high-voltage circuits, are not often discussed in literatures or researched in details. But it is essential to note that many flash failures are not due to the cell itself but rather the peripheral circuits. In general, the voltages generated by peripheral circuits might not be accurate enough to represent the correct reading level; therefore it might lead to reading errors thus failures of the cell. As shown in F. Masuka's paper "A 4-Mbi NAND-EEPROM with Tight Programmed Vt Distribution" [14] and "A NAND Structured Cell with a New Programming Technology for Highly Reliable 5-V Only Flash EEPROM" [15] , typical hot-carrier injection (HCI)

stress was conducted during program and erase cycles. Junction breakdown and punch-through voltage are in excess of 20V to withstand the stress. The reliability issues for hot-carrier stress, high-voltage oxide damage, high-field junction damage, latch-up and snapback.

Figure 4: Drain and substrate current of HV transistor before/after stress



As shown in **Figure 4** above, the drain, source and substrate currents of a high voltage transistor behave differently before and after enduring program and erase stress. In order to ensure product reliability, high-voltage transistors in peripheral circuits should be investigated carefully. However, the research conducted is on a system level where the physical behavior of each failure mechanism was not the focus of this study. A simplified model involving a Fermi function is developed to estimate the reading errors in peripheral circuits and they are shown in Section 3 below.

2.4 Random Number Generation Schemes

Random number generation has been an interesting problem in the field of mathematics and computer science for a long period of time. One might argue that true randomness does not exist in the nature but scientists keep innovating to find better algorithm or devices to generate pure random numbers.

There are two main principle methods used to generate random numbers. One use computational algorithm that produce long sequence of apparently random results, which is typically determined by an initial value called seed or key. This type of random number generated is called pseudorandom number generation. The random functions in many programming language such as C, C++, JAVA and MATLAB use this scheme where the first generated number is always the same unless a different seed is specified explicitly. This mechanism is not considered truly random because the output is inherently predictable although the repeating sequence could be extremely long. On the other hand, there is a random number generator mechanism based on physical phenomenon that is expected to be random. The physical method is most interested in the study because it would be comparable to the random number generation using the characteristics of random telegraph noise.

As early as of April 1947, RAND corporation started generating random digits with an "electronic roulette wheel", consisting of a random frequency pulse source of about 100,000 pulses per second gated once per second with a constant frequency pulse and fed into a 5-bit binary counter. Later on, there were many random physical devices or processes used to generate random numbers, including gamma ray radioactive decaying materials, quantum mechanical noises, thermal noises from resistors, avalanche noise

generated from an avalanche diode or atmospheric noise detected by radio receivers. The latest innovation came from a team of researchers from Bar-Ilan University in Israel where they were able to use optical property to design a physical random bit generator at 300 Giga bits per second, making it the fastest ever random number generator [14].

Although the theory behind physical random number generators is a pure random event in nature, the practicability or feasibility of many of these devices are quite poor. The results tend to be biased due to many unknown reasons and the randomness tends to decrease as the device degrades. This is caused by the fact that entropy sources are often fragile and fail suddenly. It is difficult to continuously perform statistical tests on these devices. As a result, many of these hardware random number generators have to be constantly monitored to ensure performance.

After proposing the probabilistic model used to characterize the behavior of flash memory failure, an application measuring the parameters of random telegraph noises is used as an innovative approach to generate random numbers from physical flash memories in Section 4. The flash memory random number generators are tested and the results are compared with current state-of-art physical random number generators.

3. Proposed Reliability Model

3.1 Introduction

The proposed model to study reliability behaviors of flash memory cell and the verification and simulation processes are presented in this section. In the first part, the three main physical phenomenal that lead to failure of flash cells are explained in details. In addition, mathematical expressions used to model the behaviors are presented. In the second section, experimental methodology and setup procedures are shown and the experimented results are analyzed and compared with simulation results perform in MATLAB. In the last section, the six parameters proposed in the model are extracted from the physical experiments. Exhaustive search and simulated annealing approach of parameter extraction are demonstrated in this part, and the actual results are compared with simulation results to ensure accuracy.

3.2 Proposed Reliability Model

Flash memory reliability problem is a complicated issue that it could be caused by many random factors such as manufacturing process variation, temperature, or its physical characteristics. However, it is essential to understand the failing mechanisms for most of the flash cells within a chip in order to improve the performance and endurance of the next generation flash memory chips. A simplified mathematical model intended to do so is presented below with three major contributing factors: stressed-induced leakage current, random telegraph noises and peripheral circuit caused reading errors.

3.2.1 Stressed-Induced Leakage Current

Stated previously in Section 2.1, leakage current increases due to a higher stress level in flash memory cells as more program and erase cycles are performed. It was concluded that the statistical distribution of the cells were well behave as an exponential function despite the difficulty to model many individual characteristics such as occasional erratic erase current. As a result, the threshold voltage as a function of time is assumed to be an exponential decay with time constant τ. It is assumed that the threshold voltage in equilibrium state is zero. The mathematical expression of this deterministic model is shown below:

$$V_{th}(t) = V_{th0} \times \exp\left(-\frac{t}{\tau_1}\right) \qquad (2)$$

where V_{th0} represents initial threshold voltage and τ represents the time constant of the exponential decay. Due to the nature of flash memory cells, after each erase cycle, the initial threshold V_{th0} is expected to be a bit different than the previous one. A bit that shows no error can be modeled with initial threshold that is far away from the tail and a large time constant τ.

3.2.2 Random Telegraph Noises

Random Telegraph Noises (RTN) consist of a sudden step-like transition between multiple discrete voltage levels at random and unpredictable times. The exact number of traps, or sudden jump in voltage level, is unknown and depends on the probabilistic model of electron tunneling. However, it is most likely that at most one trap RTN occurs during the entire endurance cycle of flash memories. It is observed that the time threshold

voltage stays in up-state and the time threshold voltage stays in down-state are two independent exponential distributed random variables with different mean time. As a result, a probabilistic model to estimate the Random Telegraph Effects on the threshold voltage is used and the mathematical expression are shown below:

$$P(t_{up}) = \frac{1}{\tau_{up\_mean}} \times \exp(-\frac{t_{up}}{\tau_{up\_mean}}) \quad (3)$$

$$P(t_{down}) = \frac{1}{\tau_{down\_mean}} \times \exp(-\frac{t_{down}}{\tau_{down\_mean}}) \quad (4)$$

Where t_{up} and t_{down} represents the time it stay in up and down states, respectively; τ_{up_mean} and τ_{down_mean} represent the mean time for up and down time distributions, respectively. The equations above indicate the probability density function for the up and down time. Monte-Carlo simulation should be used in order to incorporate the probabilistic nature of this representation.

3.2.3 Peripheral Circuit caused Reading Errors

In flash memory cells, peripheral circuits are always needed to generate various voltages for the analog nature of operations. As multi-level cell flash chips become more prevalent, different levels in floating gates only separated by a few hundred of a millivolt. Therefore, the reliability of peripheral circuits, in other words the reading error caused by it, would play a significant role in determining the error pattern of flash chips. A Fermi function is proposed to model this behavior. This is a probabilistic model and the mathematical expression is expressed below:

$$P(t) = 1 - \frac{1}{\exp\left(\alpha \times \left(V_{th}(t) - V_{fermi}\right)\right) + 1} \quad (5)$$

V_{fermi} is voltage level chosen to be exactly at the border of failure, α is the variable that controls the steepness of probability of failure, P(t) represents the probability of getting the wrong read as a function of time. A random number between the interval of 0 and 1 has to be generated for every point of time. If the generated number is greater than the actual probability, then it indicates that an error occurs. On the other hand, it shows a correct read if the random number generated is less than the probability. It is worthwhile to note that the probability of correct read would reach 1 if the threshold voltage is far away from the Fermi level.

3.2.4 MATLAB simulation

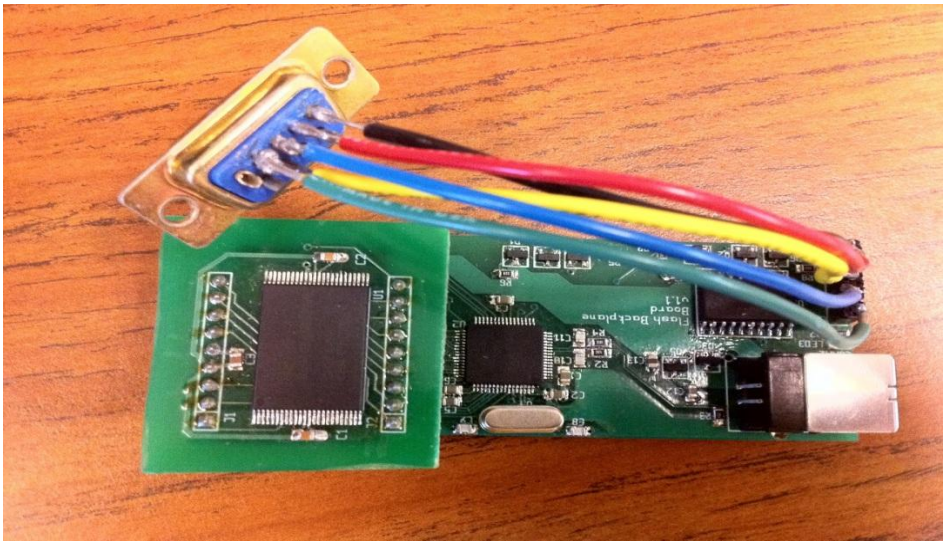
Based on the probabilistic model shown previously, MATLAB simulation is performed and used to compare and validate experimental results. The threshold voltage based on solely the effect of leakage current is computed. A set of exponentially distributed random numbers are generated using the exprnd function in MATLAB with specified up and down mean time is generated to represent the time spent on up and down states, respectively. Then a pointer is used to keep tracking the position where RTN occurs and compute the threshold voltage based on the accumulated effect of SILC and RTN. Lastly, the probability of the Fermi function is used as a decision rule to determine if an error does happen or not. Please refer to **Appendix I** for the detailed code used for simulation.

3.3 Experimental Setup and Results

3.3.1 Experimental Setup

Physical experiments to test the reliability behaviors of flash memory chips are performed. Various 4G NAND04G-B2D flash memory chips from Numonyx are used in testing. Self-made printed circuit board with a microprocessor serves as the hardware platform and the flash chips are plugged into the socket for testing. Program, Erase and Read operations are performed via instructions from the microcontroller and the results are transferred back to the PC. The detailed hardware setup is shown in **Figure 5** below.

Figure 5: Hardware setup for reliability study experiment



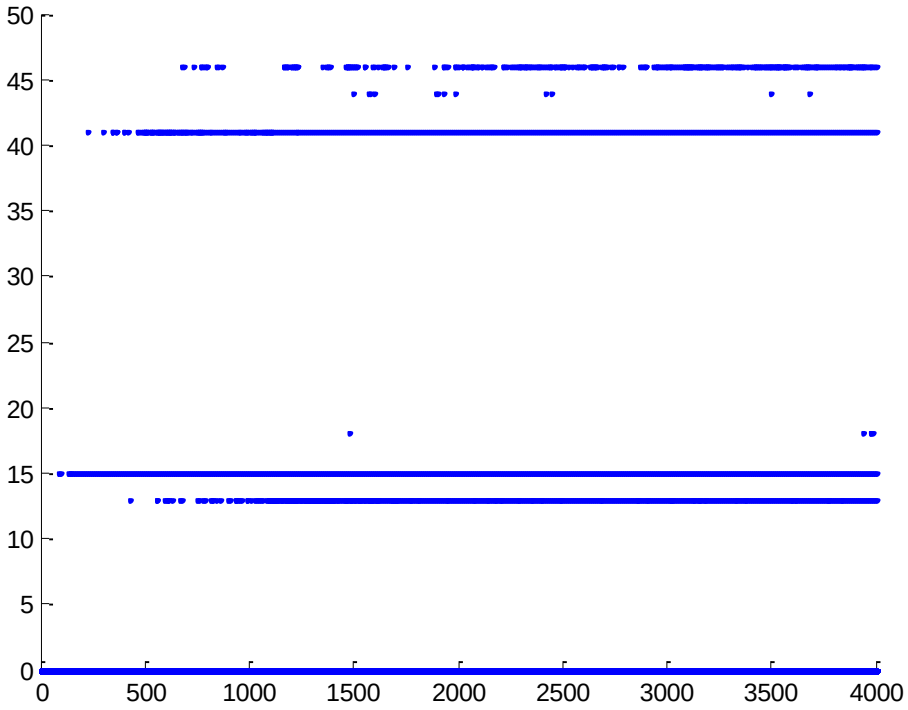
Initially, one block of the chips are stressed out with ten million program and erase cycles. This procedure is designed to wear out the chip, therefore the errors would occur more frequently and it would be more convenient to analyze the error patterns. After that, various bits are continuously read many times. A correct read is recorded as a 0 and an error is recorded as 1. A sampling frequency of 17.96 kHz is used in the continuously reading operation. This is the highest reading frequency we could achieve

with the in-house hardware. The high frequency reading is intended to reduce the aliasing in the output signal whereas the pattern of RTN would be recognized.

3.3.2 Experiment Results

Raw results received from the experiment described above are sets of binary bits where 1 indicates an error and 0 indicate a correct read. Two sets of binary results are selected and shown in **Figure 6** below. The x-axis is the number of samples and the y-axis is bit number tested in the experiment.

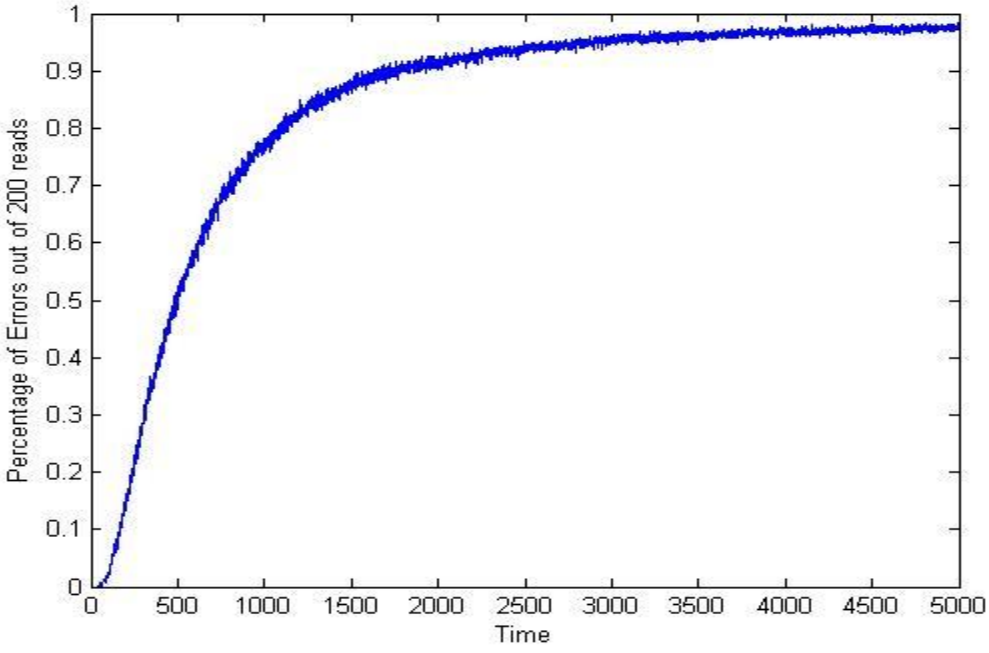
Figure 6: Raw Experiment Data (dots indicating errors)



Obviously the challenging task is to translate the binary results into numerical values of the six parameters in the proposed model where we can quantify the failing behaviors. Two different analysis tools are used in this study. The first approach involves

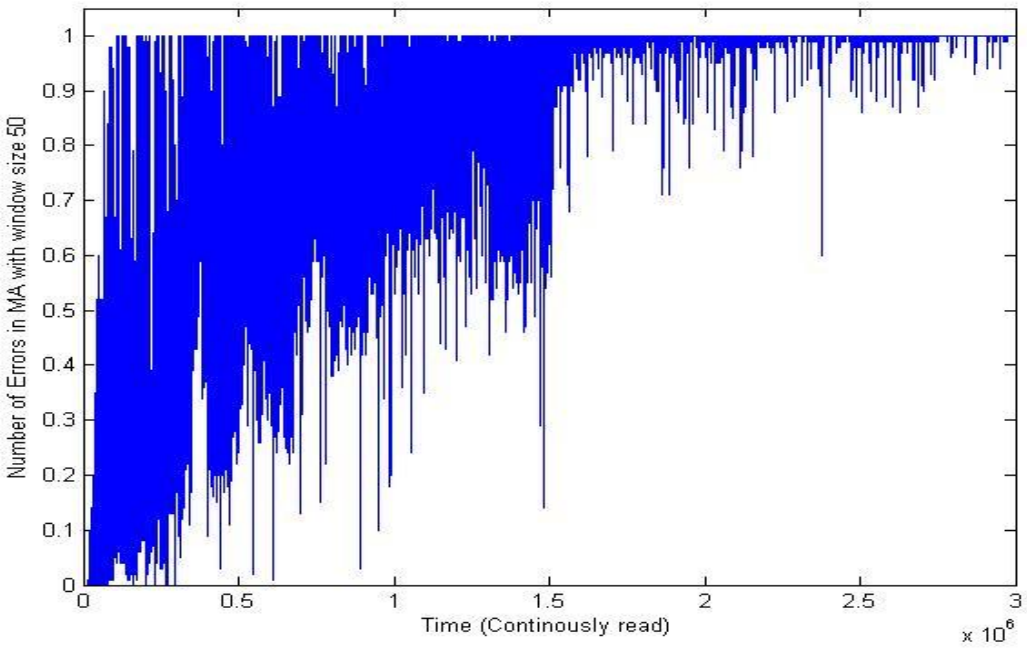
repeating the continuous reading operation 5000 times, then averaging the results of the 200 traces and considers it as the error pattern for a specific bit. This analysis comes from the idea where each trace might involve randomness based on the probabilistic model. By considering the average of 200 runs, the randomness factors from each program and erase operation would be cancelled out and the error pattern would be the end results of the model proposed previously. **Figure 7** below illustrates the result using this first approach.

Figure 7: Percentage of Error vs. Time Graph for Average Trail Analysis



The second approach is to analyze each trace of error pattern rather than the average of many runs. This is intended to illustrate the behaviors of memory cells after each program and erase cycle, thus the parameters extracted from each run could be compared and analyzed. Moving average of a window 100 is used to process the original binary sequence and it is intended to smooth out the short term fluctuation in order to find the effects of the three contributing factors proposed in the model. **Figure 8** below shows a graph of moving average of an error pattern performed in MATLAB.

Figure 8: Moving Average Error Pattern for a Single Trace



3.4 Parameter Extraction Algorithms

With the proposed model illustrated in Section 3.1 above, the difficulty is that how to translate the experimental results into numerical values for variables shown in the model. Two parameter extraction algorithms are presented in this section. The first approach uses exhaustive search mechanism where every combination of parameters is tested. The second approach uses simulated annealing algorithm to minimize the total error. A brief description of each algorithm is shown first, then the approaches are compared and the results are discussed.

3.4.1 Exhaustive Search

Exhaustive search is an algorithm that every combination of possible results is tested in order to find the best optimal result. Obviously, this approach is extremely accurate given the condition that the optimal solution is in the domain of the search. However, the major drawback of this algorithm is that the speed is extremely slow and lots of computational power is wasted. For example, given 6 parameters in the proposed model and each variable takes on 100 possible values, a total number of 10 to the power of 12 comparisons have to made in order to achieve the optimal result. It is estimated that it takes about 1 minute to complete 200,000 comparisons with an Intel Dual-core i5 2.4GHz processor. Therefore, it is required to take 10 years to complete the entire calculation! For simplicity and time efficiency, a set of 12 possible values for each variable is chosen and shown in **Table 1** below. For every combination of parameters, the probability of a correct read with a given time is calculated. The minimum mean square error criteria are used and the summation of error for the entire set of times is the objective function for this exhaustive search algorithm. The main MATLAB script for exhaustive search algorithm is shown in **Appendix II.**

3.4.2 Simulated Annealing Algorithm

The main drawback of exhaustive search algorithm is its inefficiency. Most of the time, it is not feasible to compute every combination of possible parameters. Therefore, a less computational intensive optimization scheme has to be devised. However, the performance of the new scheme should not compromise too much to compensate time

Table 1: Parameter Values used in Exhaustive Search Algorithm

Parameter Name	Minimum Value	Maximum value	Interval
V_{th0}	-0.001	-0.221	0.02
$1/\tau$	50	650	50
α	50	380	30
V_{fermi}	-0.001	-0.221	0.02
up_mean/dt	20	1220	100
down_mean/dt	20	1220	100

complexity. Simulated Annealing algorithm is the approach used in this experiment. The name and inspiration of this algorithm comes from annealing in metallurgy. It is a technology where controlled heating and cooling of material increase the size of crystals and reduce their defects as well.

As analogy to the physical process, each step in simulated annealing optimization algorithm replaces the current solution with a neighboring solution, giving a probability that is dependent on the difference between the corresponding values and a global variable K. This methodology allows the results move to a "worse" solution with a low probability, but it prevents the solution to be trapped in a local minima rather than the global minima required by the solution. The simulated annealing result will eventually reach the global minima if it is run at an infinite time. Practically speaking, the algorithm will run until the incremental error gets extremely small, or after a fair amount of computation is performed and the results are assumed to be reasonable well. This

algorithm was described by Scott Kirkpatrick, C. Daniel Gelatt and Mario P. Vecchi in 1983. [15]

The Pseudo Code of implemented simulated annealing algorithm with comments is shown below:

S = S0; e=E(s) // initial state

Sbest=s; Ebest=e; // initialize the error

While count<max_count and enew-ebest<Epson

//while loop until count reached or error small

Snew=neighbor(s); // randomly generate the neighbor

Enew=E(snew); // calculate the new error

If accept_prob>random() then // decide if it is going to move

S = Snew; e=Enew // calculate error

If Enew<Ebest then // keep track the best solution

Sbest->Snew; Ebest->Enew;

Count=count+1; // increment the count

Return Sbest

The formula used in acceptance probability is shown below:

Acceptance Probability:

If e'<= e, p=1

Else e'>e, p = exp((e-e')/K)

// K is a global parameter K=200 is chosen for this experiment

The MATLAB simulation code for simulated annealing algorithm is shown in **Appendix III** below.

3.4.3 Comparison and Results

Both exhaustive search and simulated annealing algorithm are implemented in MATLAB. The SA algorithm sacrifices accuracy of the results to improve the time efficiency of the algorithm. The difficulty of the exhaustive search algorithm is to chosen the initial set of parameters where the optimal combination lies within the range. As illustrated in Section 3.4.1 above, a maximum of 12 values for each variable can be set as a search domain. On the other hand, simulated annealing would allow a much larger search domain where many more data points can be examined. However, the difficulty with the SA algorithm is that how many sets to combination have to be run in order to conclude that it is a reasonable good result. By trial and error approach, this count is set to be 500,000 times where the incremental improvement on reducing total error becomes irrelevant.

A set of experiment is design to actually test the performance of the two algorithms. The first experiment sets the range of variables to be exactly the same (12 possible values for each parameter) for both algorithms and the accuracy of results and the computational time is analyzed. Please note that 500,000 runs are set for the simulated algorithm. The experimental results are shown in **Table 2** below.

The second experiment sets the range of variables to be all possible ranges given the physical characteristics of each variable. However, the intervals of variables in exhaustive search algorithm have to be much greater than the simulated annealing approach in order to finish running within a reasonable amount of time. The error % column records the improvement in error reduction as a percentage. The experimental results are shown in **Table 3** on next page.

Table 2: Result of the First test to compare Exhaustive Search vs. SA algorithm

Bit #	Search Error	Op Time	SA Error	Op Time	Error %
2	0.274	153s	0.286	30s	-4.4%
21	0.162	152s	0.165	31s	-1.8%
26	0.0897	151s	0.115	30s	-12.8%
31	0.0013	149s	0.0013	29s	0%
58	0.132	150s	0.145	30s	-9.8%

Table 3: Result of the Second Test to compare Exhaustive Search vs. SA algorithm

Bit #	Search Error	Op Time	SA Error	Op Time	Error %
2	0.334	148s	0.225	28s	+48.4%
21	0.262	136s	0.135	30s	+94.1%
26	0.159	145s	0.118	31s	+34.5%
31	0.0015	152s	0.0013	29s	+15.3%
58	0.162	153s	0.135	27s	+20.0%

Based on the results shown in Table 2, it is observed that the discrepancy in accuracy between the Exhaustive Search and the Simulated Annealing approach is minimal while the time consumed in SA algorithm is almost one fifth. Similarly, with smaller intervals in the range of variables in the SA algorithm, the actual result performs significantly better than the exhaustive search and it is with much shorter computation time. As a result, in order to extract parameters with a higher accuracy, it is concluded that simulated annealing approach would best fit with this study. All future experiments

and parameter extraction cases use the simulated annealing algorithm demonstrated in Section 3.4.2 above.

3.5 Result Analysis

In this analysis, 80 bits (or 10 bytes) in the given flash memory chip are tested after 10 million program and erase cycles using the proposed reliability model stated in Section 3.1. After wearing out the chip, continuous reads for 50 seconds is performed after each program and erase operation. It is observed that 20 bits does not exhibit any error patterns. There are 22 bits that exhibit somewhat error patterns, but the experimental results are inconclusive to determine the parameters using the proposed model. However, there are 38 bits that shows conclusive evidence or enough error patterns to demonstrate the validity of the model. There are 3 bits that reach "always error state" in less than 0.07 seconds. There are 7 bits that reach "always error state" in less than 1 second. In addition, there are another 5 bits that reach "always error state" in less than 10 seconds. Lastly, there are 23 bits that exhibit "always error state" within 50 seconds. Two sets of experiments are performed on the 38 bits that exhibit obvious error patterns. Simulations are performed in MATLAB to compare with experimental results. Parameters in the proposed model are extracted and analyzed in the following sections.

3.5.1 Average Trial Analysis and Results

As illustrated previously, the results received from the experiment are in binary forms which errors are recorded as 1's and correct reads are recorded as 0's. The first analysis performed is to perform the same operation 200 times then average out the error

results. This approach transforms the error pattern into numerical scales where it could analyzed despite random or erratic behaviors after each program and erase operation.

The analysis procedure is divided into three main components. The first step is to perform the experiment 200 times, collect the results, and then calculate the average of the errors in the 200 runs. The second step is to use the Simulated Annealing algorithm described in Section 3.4.2 to extract the numerical values of the six parameters. The last step involves running simulation with the parameters extracted from the previous step, then to compare the simulation results with the experimental results and conclude if it actually fits with the model described above. This procedure is repeated multiple times for the same bit to confirm the consistency of the results. The parameter extraction results for Bit 2 and Bit 21 are shown in **Table 4** and **Table 5** below where each analysis is repeated 5 times for the same bit.

Table 4: Parameter Extraction Results for Bit 2

Parameter Name	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
V_{th0}	-0.034	-0.058	-0.052	-0.055	-0.052
$1/\tau$	14.9	14.7	16.1	18.2	16.1
α	203	173	186	168	173
V_{fermi}	-0.010	-0.009	-0.013	-0.017	-0.013
up_mean/dt	1930	2050	1785	1820	2180
down_mean/dt	220	216	210	159	175
Total Error	0.274	0.287	0.207	0.163	0.242

Table 5: Parameter Extraction Results for Bit 21

Parameter Name	Trial 1	Trial 2	Trial 3	Trial 4	Trial 5
V_{th0}	-0.075	-0.080	-0.092	-0.100	-0.086
$1/\tau$	25.1	23.3	22.6	20.1	13.9
α	70	74	93	85	72
V_{fermi}	-0.034	-0.038	-0.066	-0.050	-0.050
up_mean/dt	4560	3850	3260	3520	3750
down_mean/dt	302	248	310	342	285
Total Error	0.138	0.160	0.145	0.092	0.126

The plots includes the experimental error pattern with the simulated error patter using extracted parameters are shown in **Figure 9** and **Figure 10** below.

Figure 9: Simulation vs. Experimental Result for Bit 2 (Trial 1)

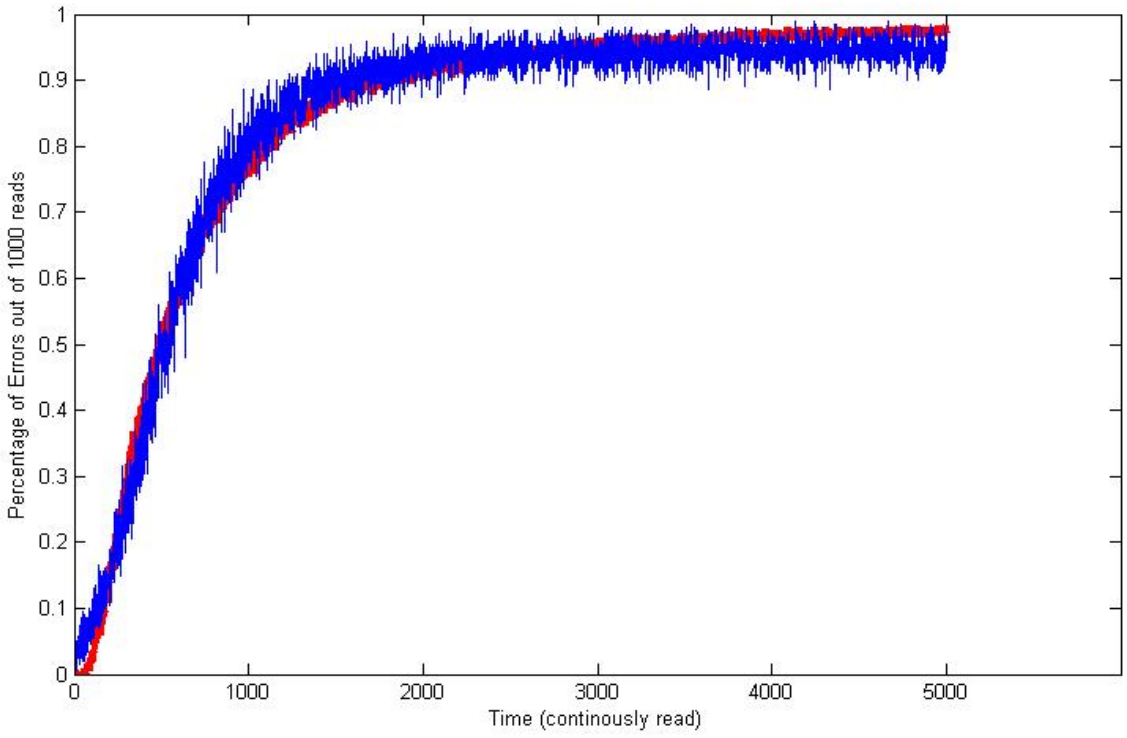
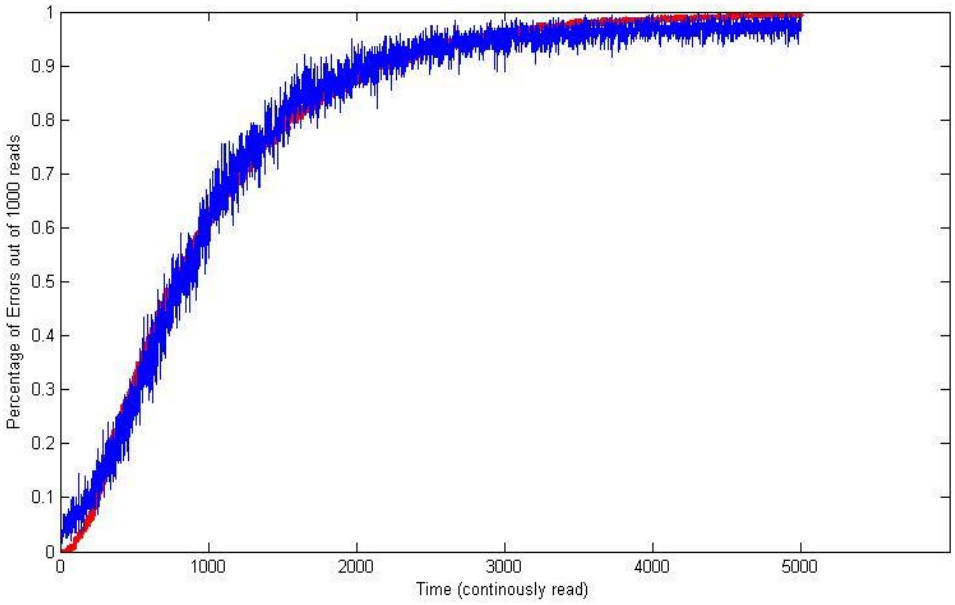


Figure 10: Simulation vs. Experimental Result for Bit 27 (Trial 1)



In the previous plots, the red colored curve shows the experimental result while the blue one shows the simulation result. It is observed that the blue colored curve has many small hops and it is the evidence of Random Telegraph Noises. Based on the analysis performed for average trial experiment for many bits, it is concluded that the extracted parameters fit extremely well with the proposed model with the total error is significantly smaller than 1, as illustrated in **Figure 9** and **Figure 10** above. In addition, as shown in **Table 4** and **Table 5**, the results obtained from multiple trials are consistent with each other with minimal discrepancy in numerical variables. These results demonstrated that the proposed model agrees with averages of experimental results and it helps validate the model as fairly accurate and consistent.

3.5.2 Individual Trace Analysis and Results

In the previous section, an average trial approach is used for parameters extraction and data analysis. However, it is believed that the random factors in erratic erase or effects from random telegraph noise might cancel out each other by averaging out the error patterns, thus not accurately characterizing the error patterns of each individual trace. In this analysis, each individual trace performed after a program and erase cycle is analyzed using signal processing techniques and the resulting patters are analyzed in details.

As demonstrated in Section 3.3.2, moving average is the first technique used to process the data. It is selected to smooth out short-term fluctuations and highlight the long-term trends of error patterns. However, the processed signal is still relatively noisy whereas the overall pattern of the signal is still difficult to observe illustrated in Figure 8. As a result, a Low Pass filter is imposed on the signal to filter out the overall shape of each signal then the Simulated Annealing Algorithm is used to extract non-RTN parameters from the pattern. The Low Pass filter filters out high frequency components of the signal, thus including the component caused by the effects of random telegraph noises.

The entire process of simulation and signal processing is performed in MATLAB. The analysis is mainly divided into four steps. The first step involves obtaining the binary results representing the error patterns, and then performing a moving average with window size using MATLAB command *output=tsmovavg(data,'s',100).* The second step is to use Low Pass filter to find the low frequency component in the signal. A Finite Input Response Filter with appropriate passing frequency is chosen for each trace. The passing frequency is selected accordingly to reflect the true low frequency component of the

signal. The filter is also designed to be with 20th order thus that it does not compromise performance with complexity of the filter. The third step is to use the Simulated Annealing approach to extract non-RTN parameters of the cell. Lastly, a simulation is run to compare the low pass filtered signal with the simulated result.

Many bits are tested for individual traces. Due to the limitation of space, only the results from bit 17 are shown in details below. This bit is representative to the majority of the bits tested and illustrates the main results achieved here. The graphs of the moving average signal (in blue) vs. the low pass filtered signal (in red) for five different traces are shown in **Figure11, Figure 12, Figure 13, Figure 14** and **Figure 15**, respectively.

Figure 11: Moving Average vs. LP filter result in Trace 1

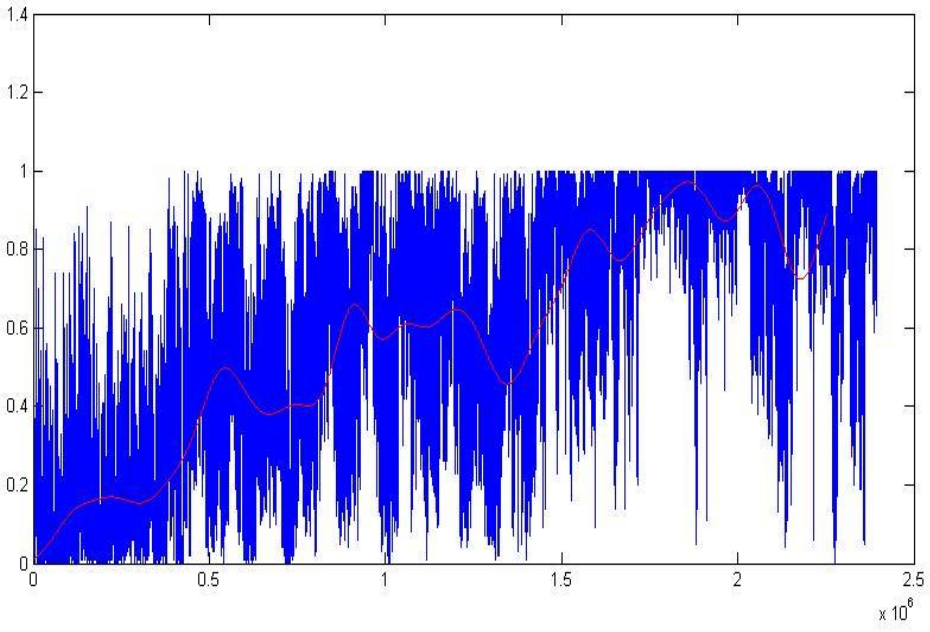


Figure 12: Moving Average vs. LP filter result in Trace 2

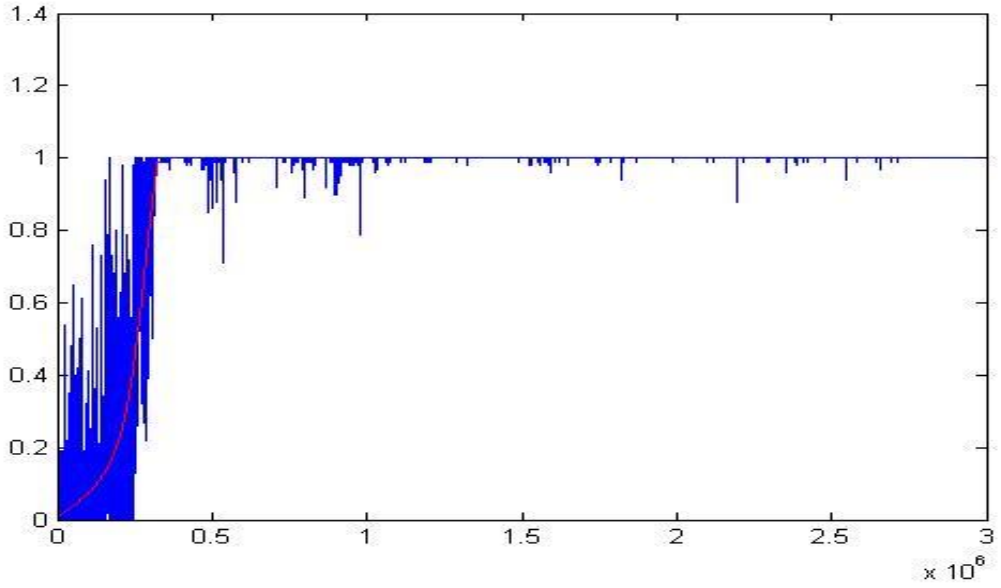


Figure 13: Moving Average vs. LP filter result in Trace 3

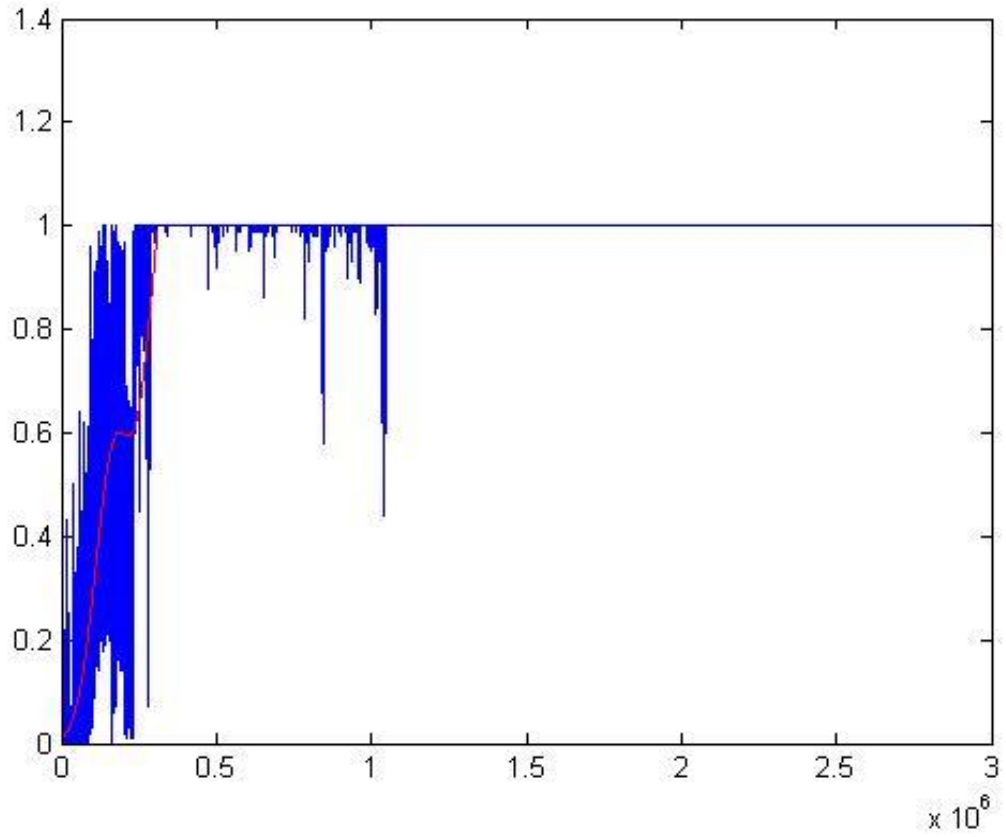


Figure 14: Moving Average vs. LP filter result in Trace 4

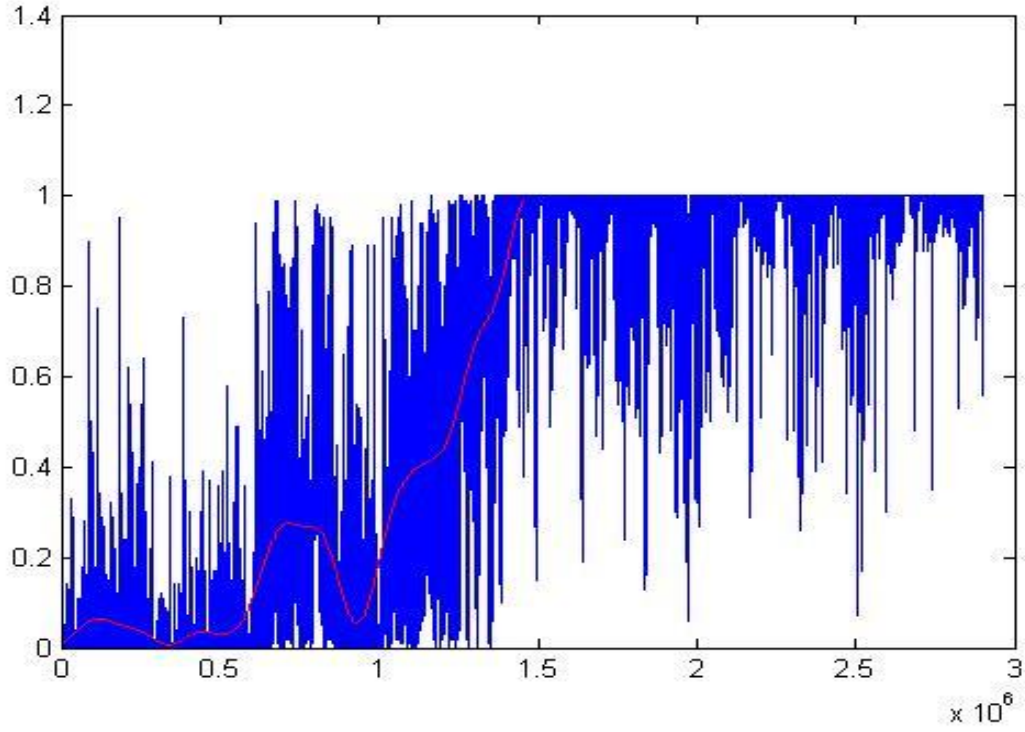
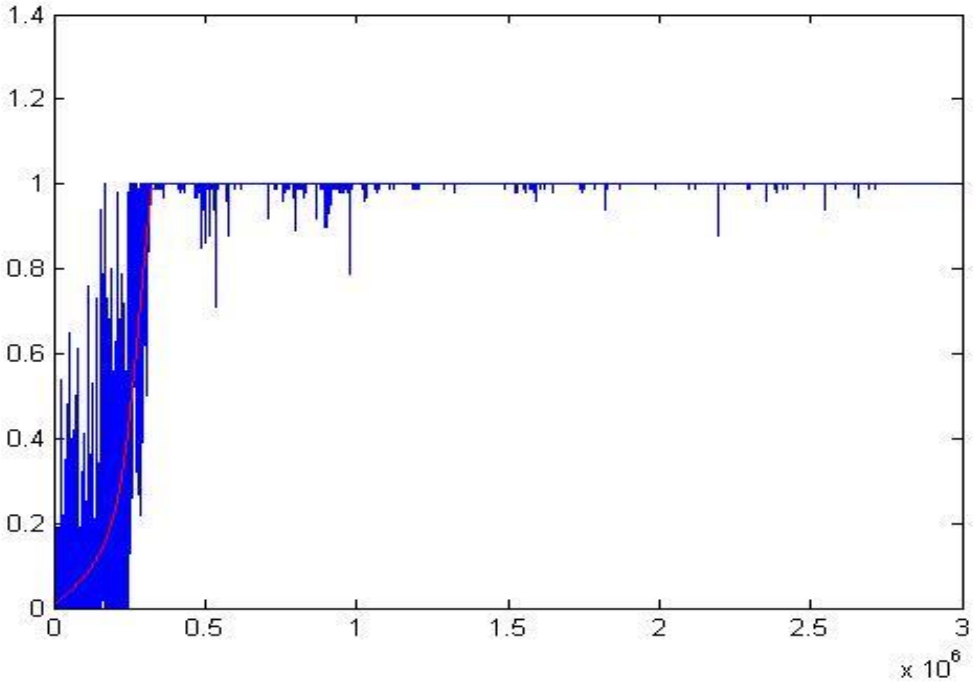


Figure 15: Moving Average vs. LP filter result in Trace 5



After careful examination of the plots shown above, it is observed that all charts follow the pattern of the proposed model. It is seen that the total number of errors generally increase as the time goes on due to the effects of the leakage current. In addition, the ups and downs of the moving average signal are caused by the contribution of random telegraph noise and peripheral circuits reading errors. However, after further investigation, the same experiment repeated multiple times has enormous discrepancy among them. For Trace 2, 3 and 5, the error pattern almost reaches complete error states with 1 million reads while Trace 1 and 4 reach complete error state much slower with about 3 million reads. Furthermore, the ups and downs in Trace 1 and 4 are much more significant compared to other traces. More importantly, the parameters extracted from the simulated annealing algorithm among different traces are quite different as well, which is illustrated in **Table 6** below.

Table 6: Non-RTN Parameters for Different Traces for Bit 17

Parameter Name	Trace 1	Trace 2	Trace 3	Trace 4	Trace 5
V_{th0}	-0.002	-0.015	-0.032	-0.003	-0.027
$1/\tau$	24.9	49.2	42.7	26.5	50.3
α	66	145	158	79	182
V_{fermi}	-0.009	-0.029	-0.045	-0.008	-0.039

Based on the parameters extracted from various traces after program and erase operations, it is observed that the results include great discrepancy among different traces. It is expected that the results should somewhat behave similarly because the results from the same bit with same stress. Bit 17 is not unique in terms of its characteristics. Many

other bits behave irrationally with different exponential decay constants for two consecutive trials. This imposes a great challenge on how to explain the behavior using physical behaviors. One obvious reason to speculate is that the initial voltage after each program and erase cycle is somewhat different. It might cause different leakage current with respect to other traces thus lead to different error patterns. Temperature could also play a factor where more experiments result higher temperature that causes erratic flash cell behaviors. In short, with the current experiment setup, the proposed model is unable to explain the vast difference in difference traces of the same bit. Model modifications or more sophisticated experiments are needed to further investigate the behaviors of individual traces.

3.6 Summary

With the proposed reliability model, the average trial experiments verify that the model fits well with experimental data and it is shown in simulation results. With the individual trace analysis, it is determined that the error pattern for each trace follows the model involving leakage current, random telegraph noise and peripheral circuit reading errors. The characteristics of each factor are obvious and they play a significant role in determining the total error patterns. Therefore, it is safe to conclude that the proposed model correctly explain the quantitative behaviors of flash memory cells in stress. On the other hand, it is a challenging and open question that if the proposed model comprehensively represent the failing behaviors of flash memory cells? According to the analysis performed for individual traces, it is difficult to account the drastic behavior of various traces of the same bit. One might assume that the average trial would cancel out

some random factors involved in individual traces thus lead to a perfect match with the proposed model. Erratic erase, manufacturing process variations and temperature effects could all be contributing factors to the failures of flash memory cells, but they are considered insignificant thus not accounted to this proposed model. In short, the proposed reliability model gives a basic and accurate representation of the major contributing factors of flash cell failures, but it is inconclusive to say that this model fully represent the failure mechanism of every single cells. Therefore, more experiments or even further improvement in the proposed model might be needed to explain the phenomenal shown in previous tests.

4. Random Number Generator using RTN

4.1 Motivation on RNG using RTN

Given the proposed reliability model, there are many interesting applications with flash memory chips in stress. Random number generators using the physical phenomenal of random telegraph noises embedded in flash memory cell is the first one that comes into mind. In addition, using the numerical values of parameters in the reliability model as a digital signature to distinguish authentic chips would also be a fascinating application that imposes unique challenges to researchers. Lastly, comprehensive understanding of flash cell failures is the ultimate goal of this study where innovative error correction codes could be designed to improve the endurance cycles of flash memory chips. This study focuses only on the first example where random numbers are generated using the characteristics within a flash memory chip.

The objective of this study is to test the feasibility of random number generator based on the physical phenomenal of random telegraph noises (RTN). Many sets of random numbers are generated, tested using a variety of techniques and then compared with other well-known random number generators. Flash memory chips are useful, cost-effective, and mass production products. It has become ubiquitous that almost everyone has a small flash memory chips in his pocket, either embedded in smart phones or MP3 players or in the form of a stand-alone portable storage device. It would be impressive and convenient that if a simple software algorithm can generate true random numbers from a piece of flash memory chip efficiently and effectively. Just imagine in the future

that an app on your iPhone generating random numbers using idle on-chip flash memory cells?

The chapter is divided into five subsections. The first part gives an introduction on random numbers. It provides the formal definition of random numbers and briefly describes the main applications that use many random numbers. The second section lists the two main types of RNG: true RNGs (TRNG) and pseudo RNGs (PRNG), and compares the advantages and disadvantages among them. The third subsection presents the detailed algorithm that generates random numbers from random telegraph noises. The fourth section presents various methods used in testing random numbers, including exploratory data analysis techniques, theoretical tests and a comprehensive test suite of binary sequence from National Institute of Standards and Technology (NIST). The random sequences of numbers generated using the RTN algorithm is tested using the above mechanism. The fifth section analyzes results and compared with other popular random number generators. It summarizes the results and proposes some recommendations on how to improve this random number generator in future research.

4.2 Introduction on Random Numbers

4.2.1 Definition of Random Numbers

Intuitively speaking, random numbers are deemed that the next generated number is unpredictable given previous numbers. For a binary sequence, the true randomness is given by this formal mathematical statement:

Let X_n be a sequence of random variables, where n = 1, 2, 3 …

X_n is a binary random variable, meaning that the possible values of X_n are 0 and 1.

If P (X_n =1 | all others) = 0.5

Or equivalently, the joint distribution of all the sequence is 0.5^N and every point of the sample space, or which there are 2^N points.

Should a sequence satisfy this definition then it can be considered random.

In addition, the random binary sequence could be transformed into integer forms under a uniform distribution. For random numbers generated between 1 and 100, they should possess the following mathematical properties:

- Uniform: If a set of random numbers between 1 and 64 are generated. The first number in the sequence to appear is equally likely to be 1, 2, 3, …, 64. Also, the i^{th} number is also equally likely to be any number within the range. The average of the numbers generated should be 32.5.
- Independence: The values must not be correlated. If it is possible to predict something about the value in the sequence, given that the previous values are known, then the sequence is not random. Thus the probability of observing each value is independent of previous values.
- Summation: The sum of two consecutive numbers is equally likely to be even or odd.

4.2.2 Applications of Random Numbers

Random Numbers have been used in a variety of fields in everyday lives. They include but not limited to applications in cryptography, sampling, simulation, gaming, decision making, aesthetics, and other areas where producing an unpredictable results are

desirable. How random numbers are used in the above fields is briefly described in subsections below.

1) Cryptography

Cryptograph is the science of turning meaningful sequence of information into apparently random noise in such a way that only a key-holder can recover the original data. The main objective is to preclude an adversary from gaining advantages through knowing what the message says. Sequences that are hard to predict unless the mechanism generating them is known are required to achieve the goal. The heart of all cryptographic is the generation secret and unguessable numbers – random numbers. Cryptography has been used in many areas including securing e-commerce around the word, protecting private communication over internet, not to mention used by all governments to protect national secrets.

2) Sampling

It is often difficult to exanimate every possible case in real world situations. With random sampling, it always provides insights to a typical behavior of the system. Sampling with random numbers gives each member of the population an equal chance of getting selected, avoiding the problem of biases. In both academia and many industries, random numbers are used in selection by researchers around the world.

3) Simulation

Simulation is the re-creation of complex phenomenal, environments, or experience, allowing people to have new understanding about the theory behind it. Random numbers are often required during the simulation process. For example, in nuclear physics, particles are simulated subject to the phenomenal of random collisions.

In addition, in the field of operation research, engineers have to simulate the rate people come to a shopping mall, which is random in nature. Increasingly sophisticated simulation studies require more involved random numbers and the results sometimes become more sensitive to the quality of underlying random numbers.

4) Gaming

Many games involve random numbers and chances, including rolling a dice, shuffling deck of cards, spinning roulette wheels and others. Randomness is obviously central to games of chances and vital to the gaming industry. As online casinos or online poker games become prevalent worldwide, the game provider certainly hopes to generate the true random numbers to protect their profit margin and the industry have increasingly becoming a huge consumer of random numbers online.

4.3 Types of Random Number Generators

There are two main types of Random Numbers Generators (RNGs). The first one is called a True random number generator based on physical randomness effects. The second type is called pseudorandom generators. The main difference between the two types is that TRNGs sample a source of entropy whereas PRNGs use a deterministic algorithm to generate random numbers. A short description of the two schemes is shown below and they are compared in a later subsection.

4.3.1 True RNGs

A true random number generator uses a naturally occurring source of randomness to generate random numbers. It samples the source of entropy and then usually processed

through a computer to generate a sequence of random numbers. True RNGs should not be taken as complete random because of its name. The entropy usually consists of some physical quantities, such as atmospheric noise from radio, thermal noises from a semiconductor diode, frequency instability of a free running oscillator or the time elapsed between the emissions of particles in radioactive decays.

4.3.2 Pseudo-RNGs

Hungarian-American mathematician once said "Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin". However, pseudorandom generators (PRNG) indeed use deterministic algorithms and it is more widely used compared to TRNGs. Pseudo random numbers do not depend on a source of entropy and they are not strictly random. They use a mathematical algorithm to compute random sequences. If the algorithm and the seed are known, then the sequences of random numbers are predictable. The objective with PRNGs is to generate sequences that behave like random. The output sequences of many PRNGs are statistically indistinguishable from completely random sequences. Ironically, PRNGs often appear to be more random than numbers generated from TRNGs [16]. However, in theory, the maximum length of sequences produced by all those algorithms is finite and these sequences are reproducible, and thus can only be random in some limited sense [17].

4.3.3 Comparison between TRNGs and PRNGs

Both True Random Number Generators and Pseudorandom Generators have their advantages and drawbacks. Each random number generator scheme is suitable for a particular application. This study mainly focuses on TRNG using the characteristics of

Random Telegraph Noises in Flash memory cells. **Table 7** below lists the advantages and disadvantages of TRNG. It is assumed that the numbers generated from TRNG are completely random.

Table 7: Advantages vs. Disadvantages of TRNGs

Advantages	**Disadvantages**
No periodicities	Slow and Inefficient
No dependencies present	Sequence are not reproducible
No predictability based on prior knowledge	Cumbersome to install and run
High level of security	More expensive
Conceptually random, not deterministic	Possibility of manipulation

4.4 Algorithm Description

4.4.1 Partial Erase Operations

Random telegraph noises are sudden step-like jumps in threshold voltages. In past literatures and experiments, it is observed that the times threshold voltage spent in up or down states in threshold voltage are two exponential distributed random variables with different mean values. The key points in the algorithm to generate random numbers is to accurately capture the one-step transition RTN behaviors and then transform the exponential random variable into an uniformly distributed random variable, where a sequence of random numbers are generated based on the probability density function. This might seem a relatively straight-forward procedure but it does impose unique challenges to capture the RTN behavior. Some of the difficulties include the fact that one

has to select the bits that have the best RTN behavior, and to determine on the methodology to process the original error pattern and how to determine and tabulate the time spends in up or down RTN states.

With millions of program and erase cycles performed in the experiment to study the reliability model, it is determined that the error signal are too noisy in many bits tested. It is concluded that it would be extremely difficult to single out the one-trap RTN behaviors to accurately model the exponentially distributed random behavior with such a high stress level. As a result, one innovative approach to use partial erase is devised and found effective for modeling the RTN parameters. The microcontroller that controls the operation of the flash memory chips has an "Abort" operation. Using the "Abort" command during the erase operation would allow the flash cell to perform a partial erase operation. Physically speaking, erase operation represents a decrease in threshold voltage from a relatively high point to a relatively low point. The partial erase allows the threshold voltage not to go back to the low value but a range in the middle. This procedure makes the flash chips more prone to errors with less imposed stress. By trial and error, experiments show that the RTN phenomenal occurs with as little as 1000 program and erase cycles and the processed signal behave well to form a one-trap step-like signal. This is suitable to extract the parameter for RTN thus generating random number based on the exponentially distributed random variable. The specific bits are selected manually to be the sources for testing RTN parameters and thus generating random numbers.

4.4.2 Feedback Algorithm

As described above, accurate generation of random numbers require microprocessor to know the exact point to stop during an erase cycle. This point should be best suitable to generate one-trap RTN signal with long duration. It is essential that the aborted threshold voltage is not too close to the full erase voltage where almost no error will occur for a long period of reads. On the other hand, the threshold voltage should not be close to program voltage where many errors would occur. This leads the signal to be extremely noisy where one-trap RTN model would not apply. A simplified feedback loop algorithm is used to choose the proper stop point during erase cycle. Once a stop point is reach, continuous read is performed for thousands of times. Then the moving average of the signal with window size 50 is used as criteria to determine if a proper stop point is reached. The feedback algorithm is written in microprocessor's code in C language. The main component of this feedback algorithm is shown below:

```
if (tot_max-tot_min>8 && tot_min<15)
        break;
else if (tot_min>=15)  //erase too little, too much error
        terase=terase+1;
else if (tot_max<8)  //erased too much, few error
        terase=terase-1;
else
        break;
```

By trial and error, the numerical values in moving averages are selected. It is required that the difference between maximum error and minimum error have to be in excess of 8 in order to distinguish the two trap levels. In addition, the minimum errors have to be less than 15 in order to prevent too many errors in the signal. The above feedback algorithm is used in following experiments to determine the proper "abort levels".

4.4.3 Signal Processing

After picking the best point to abort the erasing operation with the feedback algorithm, the microprocessor performs a continuous read operation of 2 million times and the error patterns are collected. The reading and recording process take approximately 4 to 5 minutes to complete. The number of consecutive reads is seek to be as large as possible in order to generate as many random number as possible from each P/E cycles. However, as leakage current caused by stress level increases as a function of time, the collected error pattern does not behave well as a step-like function. This leads to the problem that too many reads would negatively impact the signal thus 2 million reads is properly selected as well.

With the recorded binary error patterns, the signals are analyzed before further processing. It is observed that some of the raw data are excellent where errors are clustered together as illustrated in **Figure 16**. These signals are considered well behaved and no more signal processing is required. The frequency spectrum of the data is plotted in **Figure 17** and it clearly shows a 1/f behavior and agrees with theory. On the other hand, some signals are a bit noisy where moving average with window size 50 is analyzed. They are shown in **Figure 18** and **Figure 19**. Noise cancelling techniques are utilized in order to smooth the curve in order to approximate the step behaviors.

Figure 16: Raw Experimental Data (Good Bit, no processing required)

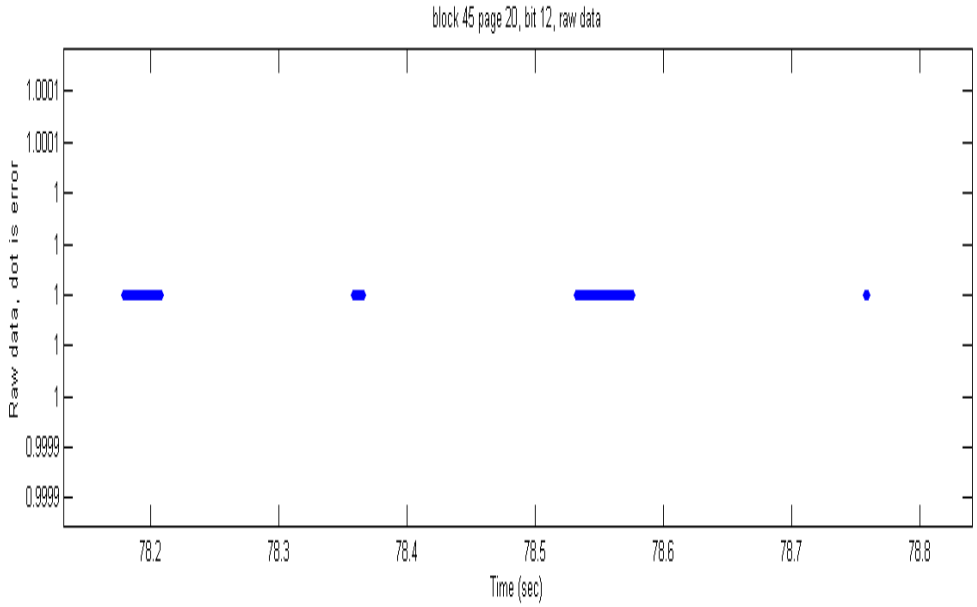


Figure 17: Frequency Spectrum of raw data (Good Bit, no processing required)

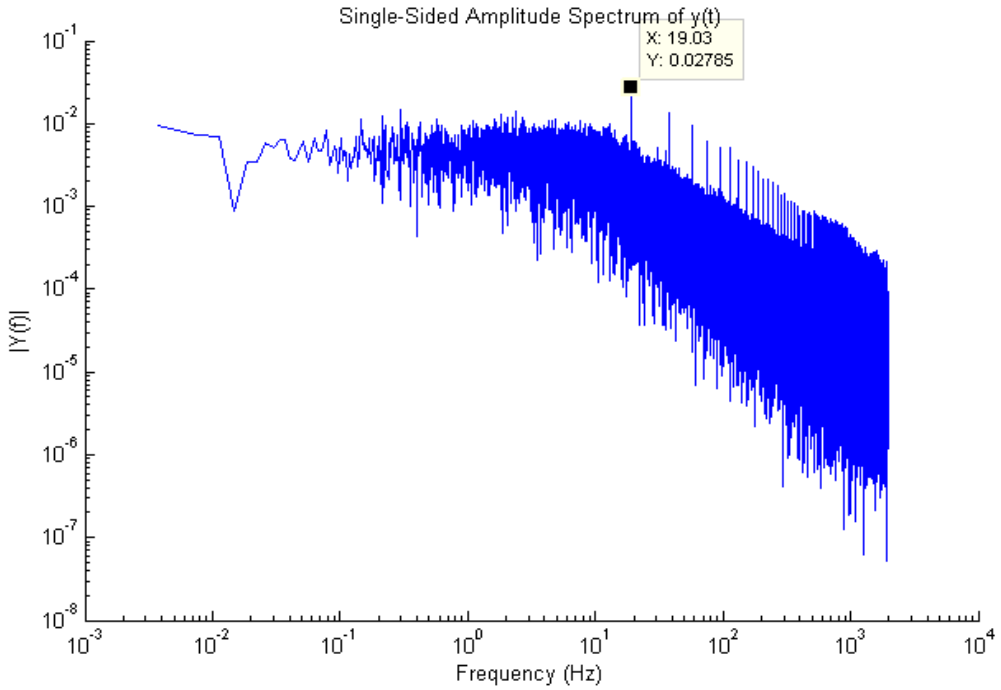


Figure 18: Moving Average of Processed Signal

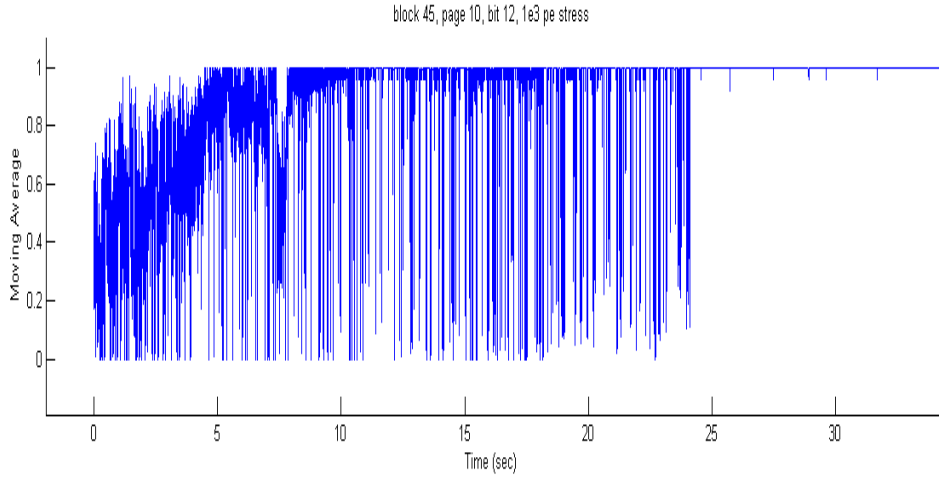
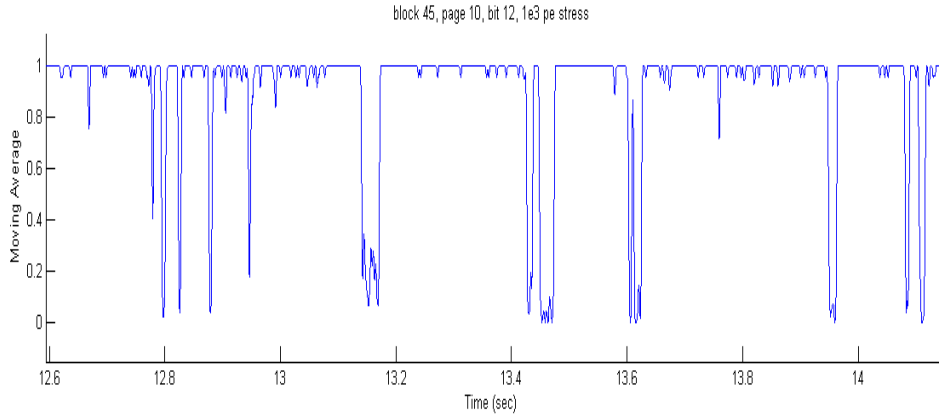


Figure 19: Enlarged Moving Average of Processed Signal



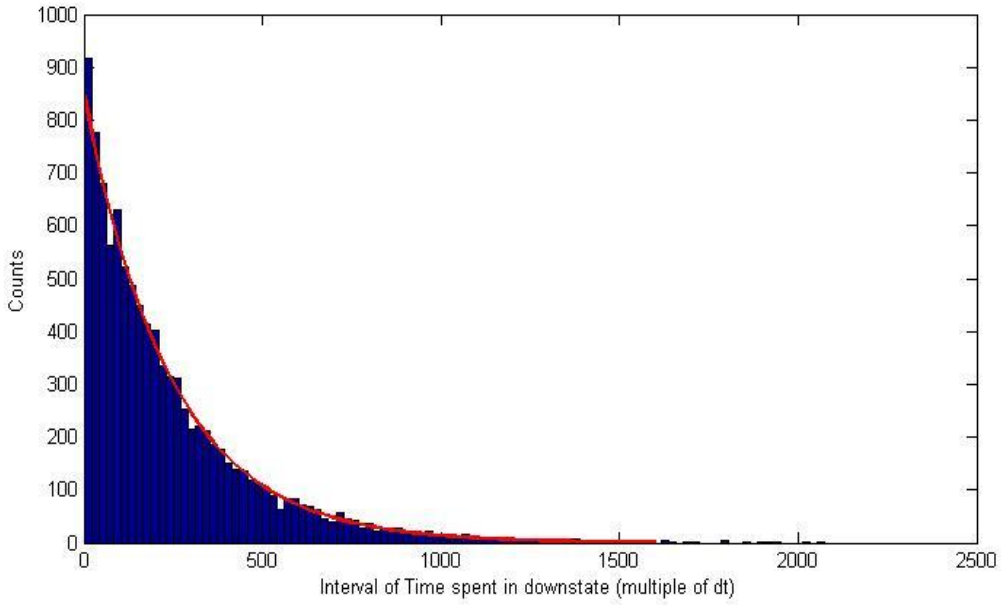
4.4.4 Transformation and Generation

After signal processing, the next step is to tabulate the time spent in up states or down states as a sequence of numbers. These times are known to behave as an exponential distributed random numbers in theory. The probability density function of an exponential random variable is shown below:

$$f(x,\beta) = \beta \exp(-\beta x) \quad if\ x \geq 0 \qquad (6)$$

where x is a random variable and represents the time spent in up/down state, β is a parameter of this density. The mean of an exponentially distributed random variable is $\frac{1}{\beta}$. Given the sequence of random variables, the mean of these numbers could easily be calculated. Thus the parameter β is based on the measurements of up/down state times. 10000 values of times spent in down states of a specific bit are collected and plotted in the histogram with 100 bins below in **Figure 20.** The histogram is then fitted with an exponential function shown in red. β is estimated to be 0.00413 and it corresponds with a mean of 242.13.

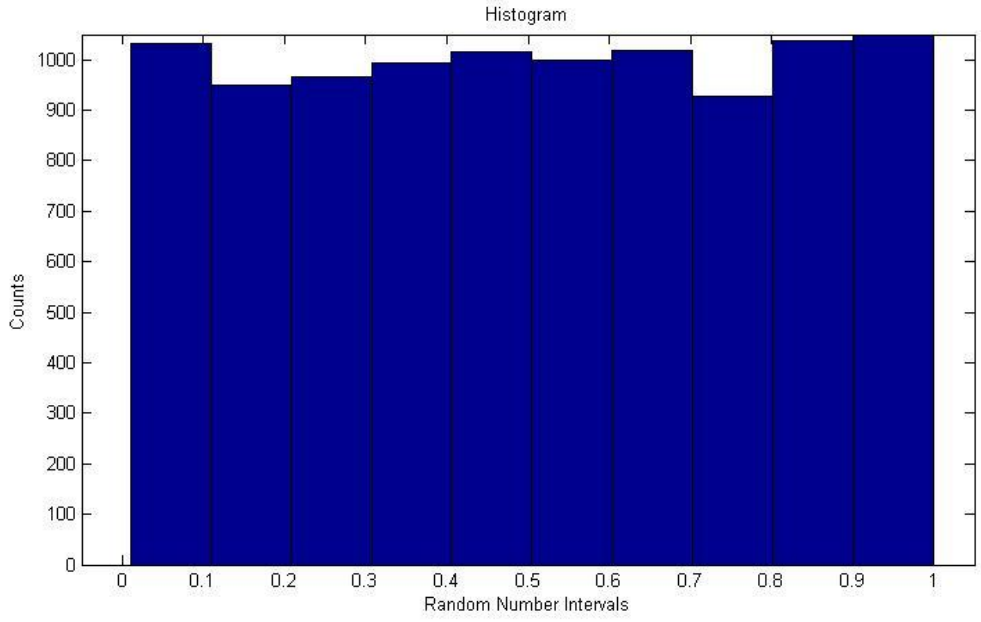
Figure 20: Histogram of Time Spend in Down States



In addition, it is known in theory that a transformation between uniformly distributed random number and exponentially distributed random number exists. Assume that Y is a uniformly distributed random number on the interval between 0 and 1, or commonly known as a true random number, the transformation $X = -\ln(y)/\beta$ would leads to an exponentially distributed random number. A simple algebraic manipulation

would yield the result that a perfect random variable Y between interval of 0 and 1, or uniformly distributed in the intervals, would exist using the following expression given a perfect exponentially distributed X. The histogram of a set of translated data is shown in **Figure 21** below.

Figure 21: Histogram of Transformed data



With the above transformation, the sequence of time spent in up/down states is translated into a sequence of random numbers between 0 and 1. This numbers would represent the outcome of this proposed random number generator. Once the total number runs out, a new program and erase cycle with abort would be implemented to generate the new set of random numbers. In theory, the random numbers produced are perfectly random subject to measurement errors given the fact that the RTN behaviors in the chip are truly random.

4.5 Test of Randomness and Analysis

With the random sequences produced using the algorithm described above, it is essential to test the performance of the RNG. Three different types of tests are used in this study: exploratory data analysis, statistical tests and industry standard binary sequence test suites. Each of the three approaches is described and the results are shown below. For the purpose of EDA and Statistical tests, the generated random numbers in the interval between 0 and 1 are times 64 to represent random integers from 1 and 64. This technique is used to help visually the data. For NIST test suites, the numbers are translated in binary sequences for testing, where number below 0.5 represents 0 and numbers greater than 0.5 represent 1. The performance testing of this RNG is still at the beginning stage. This section is mainly focuses on testing a set of 10000 random numbers generated from a specific bit's down time distribution at the beginning of the endurance cycle.

4.5.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing data for the purpose of formulating hypotheses worth testing, complementing the tools of conventional statistics for testing hypothesis. It is always necessary to conduct exploratory data analysis on a data beset before more formal tests are applied. EDA techniques are usually graphical and they are especially helpful in providing a visual exploration of the data. Summary statistics are used to give an overview of the main parameter of the data. These values are: the count (number of values in the sequence), the mean (average), the median, the maximum and the minimum value. The P (even)

demonstrates that the summation property of the numbers holds and that the probability of the sum of two consecutive numbers given in the data set is 0.498, which is extremely close to the theoretical value of 0.5.

Table 8: Summary Statistics for Testing Sequence

Count	Mean	Median	Maximum	Minimum	P(even)
10000	32.74	33	64	1	0.498

There are many EDA techniques available and most of them are graphical in nature. Four of them are selected to provide a visual measure of randomness of the sequence. They are Run Sequence Plot, Lag Plot, Histogram and Autocorrelation Plot. It is important to note that the EDA does not prove the randomness of the data by itself but it highlights pattern, outliers and apparent non-randomness relationship and bias. Any irregularity that is identified can be further investigated when conducting statistical and empirical tests.

1) Run Sequence Plot

   The Run Sequence plot is a graph of each observation against the order it is in the sequence. **Figure 22** below is the run sequence on a set of 1000 number. It is observed that the plot fluctuates around 32, which is the expected mean of this sequence. In addition, the fluctuations in the plots appear random and there is no upward, downward or cyclical trend evident in this graph.

2) Lag Plot

   The Lag Plot is a scatter of each observation against the previous observation. This plot is most useful in detecting outliers. Outliers should always be examined and deleted from the dataset if necessary before analysis carry out. If there are many outliers in the data set, it is an indication that the random number generator could

have flaws. The Lag Plot of this data set is shown in **Figure 23** below. It shows no outliers

Figure 22: The Run Sequence Plot of Tested Sequence

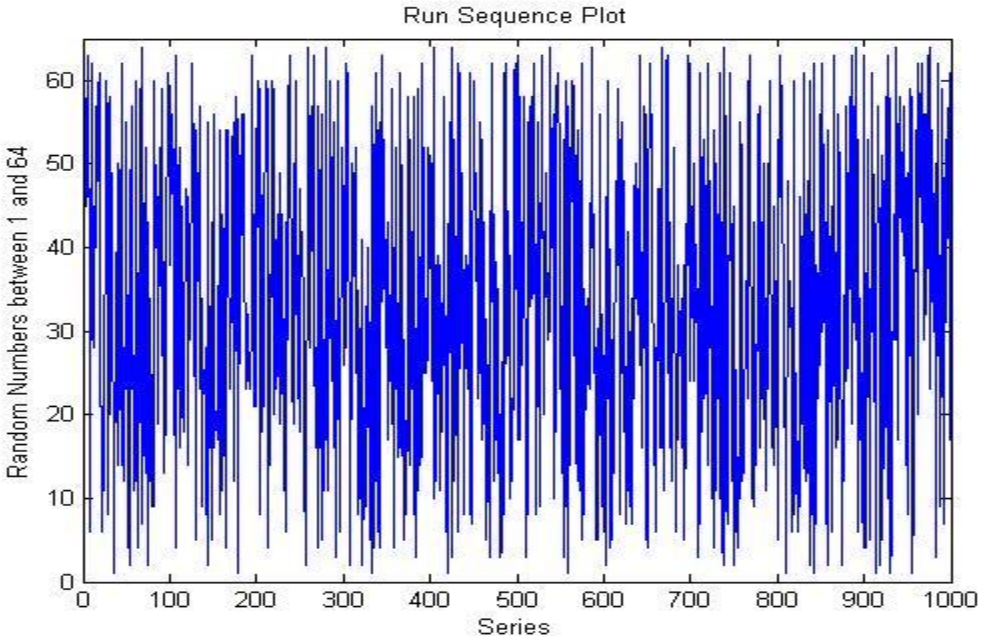
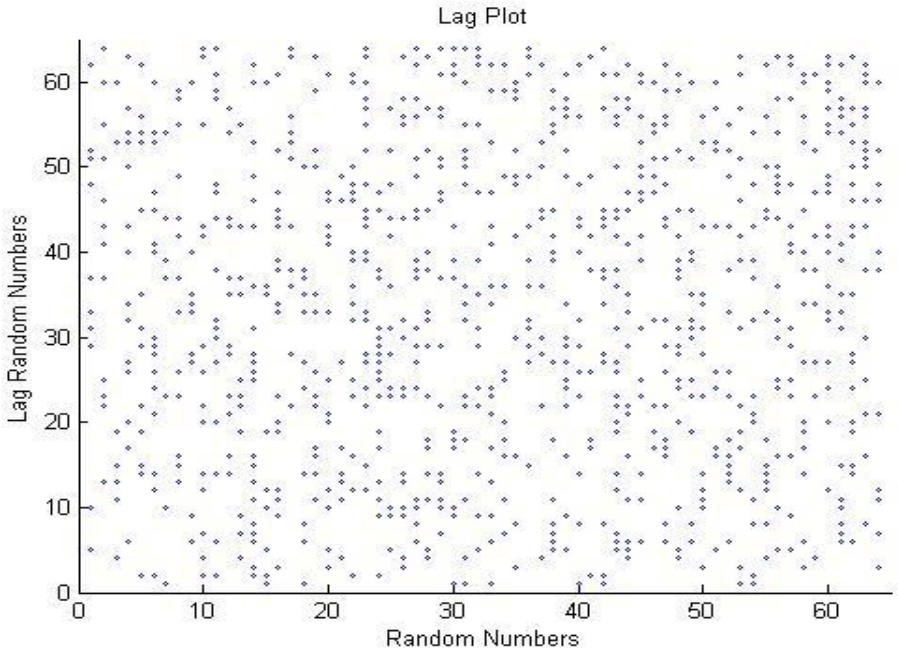


Figure 23: The Lag Plot of Test Sequence

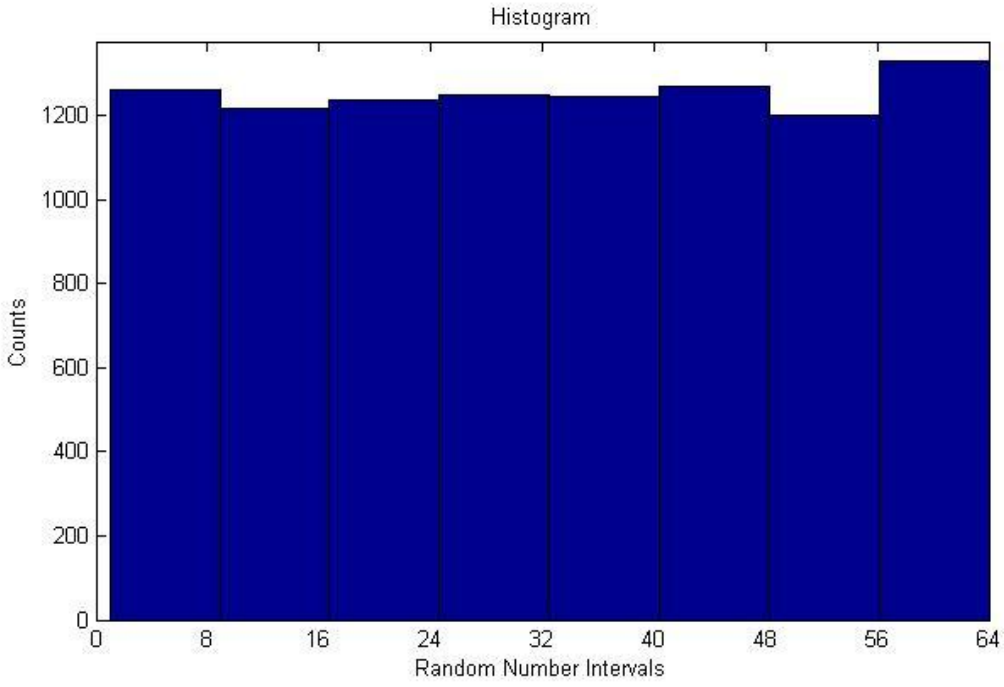


and the data are spread evenly across the plane. This is a good indication that the generated sequence is truly random.

3) Histogram

The histogram represents a frequency distribution and in this case it shows the count of observations that occurs in each sub-interval. Therefore, it is expected that almost the same number of observation lies in each bin. As shown in **Figure 24** below, approximately 1250 observations fall into each interval. The histogram confirms the property of uniformity for the sequence.

Figure 24: Histogram

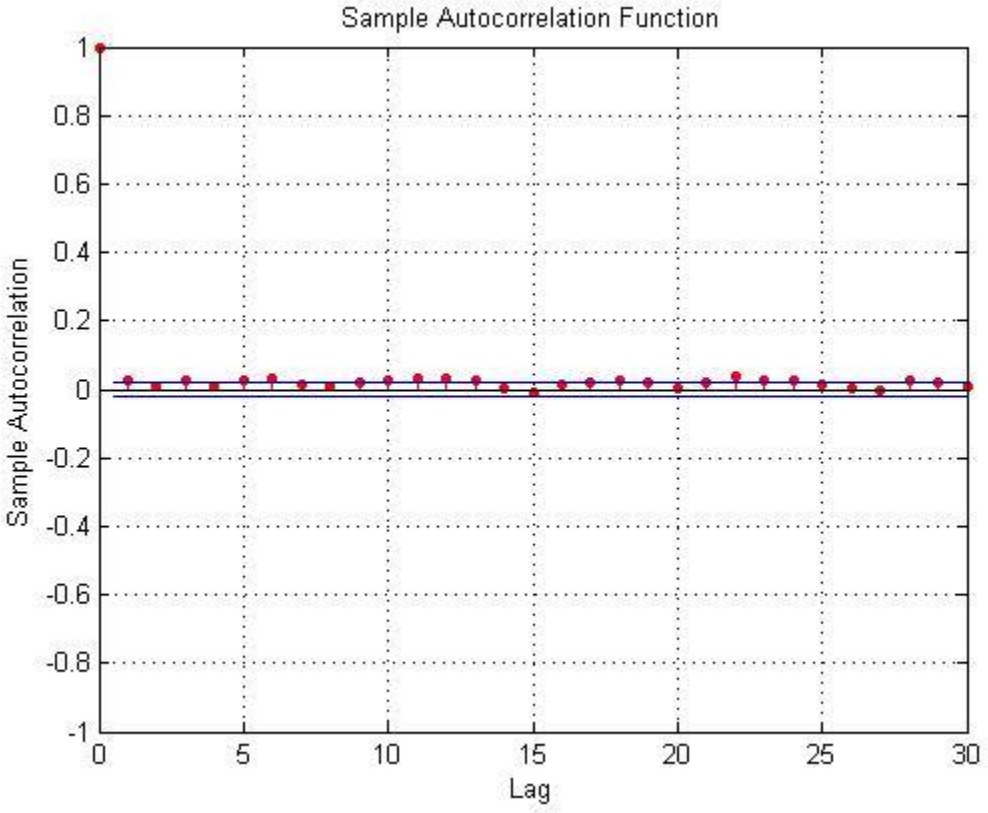


4) Autocorrelation Plot

Autocorrelation is the correlation of a variable with itself over successive time intervals. It occurs when an observation is somehow determined by proceeding observation. The x-axis is the lag between measurements and the y-axis shows the autocorrelation coefficient. The autocorrelation plot is shown in **Figure 25** below. It

is observed that all the values are in control (inside the blue line) and the correlations are extremely small. It is found that the autocorrelation is bounded by 0.02, which is well below the key threshold of 0.1. This demonstrates that there is almost no dependence in successive observations it demonstrates the property of independence.

Figure 25: Autocorrelation Plot



In summary, the exploratory data analysis shows to support the hypothesis that the numbers are random. The summary statistics illustrates the size of the dataset and provides a reasonable evidence of randomness. The mean of 32.74 agrees with the property of uniformity and it is also confirmed in the histogram. The low autocorrelation bound validates the properties of independence. In addition, the run sequence and the lag plot dos not highlight any outliers and inconsistencies in the data. The randomness of the sequence is further tested in statistical and empirical tests outlined below.

4.5.2 Theoretical Tests

Theoretical tests are primarily concerned with theories or hypotheses rather than practical considerations. The tests are formulated to test a null hypothesis. For the purpose of this test, the null hypothesis under test is that the sequence tested is random. Associated with this null hypothesis is the alternative hypothesis, which implies that the sequence is non-random. For each applied test, a decision or conclusion is derived that either accepts or rejects the null hypothesis. Three theoretical tests are selected to test the sequence: The Chi-Squared Test, the Test of Runs Above and Below the Median Test and the Reverse Arrangement Test.

1) The Chi-Squared Test

   The Chi-Squared test focuses on the property of uniformity of the sequence. The test is aimed to see if the observed frequencies in each class are significantly different from those that could be expected. Observed and Expected Frequency for the Chi-Squared Test on the sequence of 10000 numbers is shown in **Table 9** below. The deviations are shown in **Figure 26** as well. The procedure of this hypothesis testing is shown below:

   H_0: The numbers follow a uniform distribution

   H_a: The numbers do not follow a uniform distribution

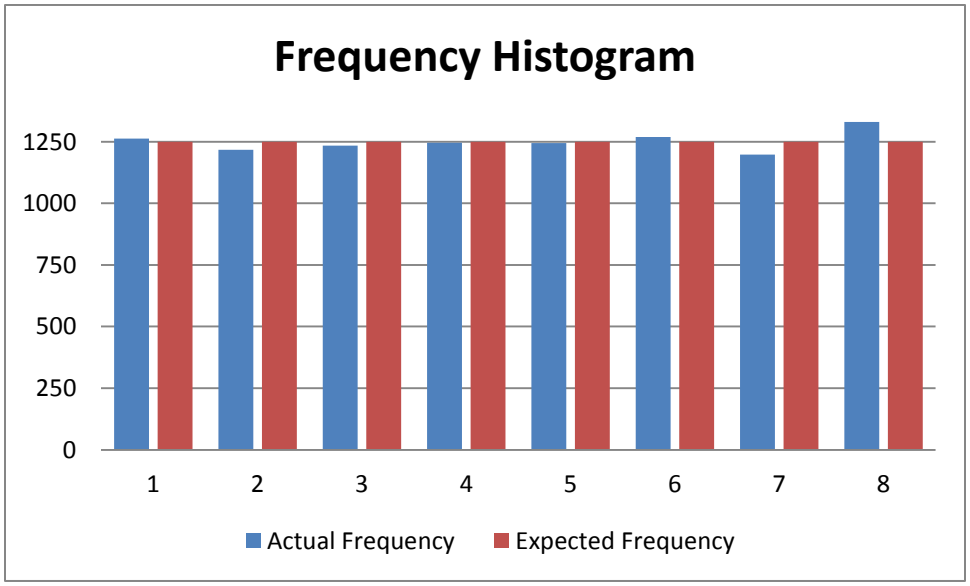
   Test Statistics: $x^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}$ where k is the number of categories

$$x^2 = \sum_{i=1}^{8} \frac{(O_i - E_i)^2}{E_i} = 8.80$$

Table 9: Observed and Expected Frequency for the Chi-Squared Test

Category	Observed (O_i)	Expected (E_i)
0.100	1262	1250
0.200	1217	1250
0.300	1234	1250
0.400	1246	1250
0.500	1244	1250
0.600	1269	1250
0.700	1198	1250
0.800	1330	1250
Total	10000	10000

Figure 26: Frequency Histogram (Expected vs. Actual)



Level of Significance: $\alpha = 0.05$

Critical Value: $x^2_{0.05,7} = 14.07$ (Critical value with 7 degree of freedom)

The test statistics is less than the critical value so the null hypothesis is accepted at 5% significant level. The resulting numbers fit into a uniform distribution and the RNG passes this test.

2) Test of Runs Above and Below the Median Test

The Test of Runs Above and Below the Median Test looks at the order of numbers in the sequence to determine if the order is random or attributable to a pattern in the data. A run is defined as a set of consecutive numbers that are either all less than or all greater than the median value. The total number of runs, u, the total numbers in the sequences greater than the median, n1, and the total numbers less than the median n2 of the sequence are collected. Then u is distributed with the following mean μ_u and standard deviation δ_u

$$\mu_u = \frac{2n_1n_2}{n_1+n_2} + 1 \qquad \delta_u = \sqrt{\frac{2n_1n_2(2n_1n_2 - n_1 - n_2)}{(n_1+n_2)^2(n_1+n_2-1)}}$$

The sampling distribution of u can be approximated to normal distribution if n_1 and n_2 are sufficiently large. In this case, it is assumed that u has a normal distribution. The hypothesis that the sequence is random cannot be accepted when the test statistics z is less than $z_{-\alpha/2}$ or exceed $z_{\alpha/2}$ where

$$z = \frac{(u \pm 0.5) - \mu_u}{\delta_u}$$

The ± 0.5 is a continuity correction included to incorporate values of u less than and greater thenμ_u. -0.5 is used when $u > \mu_u$and +0.5 is used when $< \mu_u$.

The summary Statistics for the Runs Test are shown in **Table 10** below.

Table 10: Summary Statistics for the Runs Test

u	n_1	n_2	μ_u	δ_u	z
4923	4887	4959	4922	49.61	0.01

H_0: The sequence of numbers are generated in a random order

H_A: The sequences of numbers are not generated in a random order

Test Statistics:

$$z = \frac{(u - 0.5) - \mu_u}{\delta_u} = \frac{(4923 - 0.5) - 4922}{49.61} = 0.01$$

Level of Significance: $\alpha = 0.05$

Critical Value:

$z_{-\alpha/2} < 0.01 < z_{\alpha/2}$ => $z_{-0.025} < 0.01 < z_{0.025}$ => $-1.96 < 0.01 < 1.96$

The test statistics z=0.01 lies between ± 1.96. Therefore, the null hypothesis is accepted at the 5% significance level. There is no evidence to suggest a bias in the sequence of generated random numbers. The result of this test confirms that the random patterns identified in the EDA test are indeed random.

3) Reverse Arrangement Test

The Reverse Arrangement Test focuses on the detection of biases and monotonic trends in the sequence of generated random numbers. Take N observations from a random variable X, denoted by xi where I = 1, 2, 3 … N. Then count the number of times that $x_i > x_j$ for each i<j. Each such inequality is called a reverse arrangement. The total number of reverse arrangements is denoted by A.

From the observations $x_1, x_2, x_3, \ldots, x_n$

Let $h_{ij} = \begin{cases} 1 \; if \; x_i > x_j \\ 0 \; otherwise \end{cases}$

Then $A = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} h_{ij}$

If the sequences of N random numbers are independent observations on the same random variable, then the number of reverse arrangements, A is a random variable with the following mean and standard deviation.

$$\mu_A = \frac{N(N-1)}{4} = 24997500 \qquad \delta_A = \sqrt{\frac{N(2N-1)(N-1)}{72}} = 166654$$

Test Statistics: A=25135236, or z=0.8265 (z-score)

Level of Significance: $\alpha = 0.05$

Critical Value:

$z_{-\alpha/2} < 0.8265 < z_{\alpha/2}$ => $z_{-0.025} < 0.8265 < z_{0.025}$ => $-1.96 < 0.8265 < 1.96$

Based on the above experimental data, the null hypothesis is accepted at 5% significant level. The test shows that there is no evidence that the data has monotonic trends. This supports the assumption that there is no biased in generated random numbers.

4.5.3 Comprehensive NIST Test Suites

With Exploratory Data Analysis and Statistical Tests shown above, it is still inconclusive to show the randomness of the sequence. In the past few decades, many tests suites, including empirical tests, have been developed for testing randomness in a binary sequence for the purpose of cryptography. Here are the descriptions of the three most significant test suites

1) Knuth: Donald Knuth's (Professor from Computer Science Department) book "The Art of computer Programming (1st edition, 1969)" [18] is the most quoted reference in statistical testing for RNGs in literature. Although it was a standard for many decades, it appears to be outdated in today's view. He fails to mention cryptographic applications and perhaps it was not as important as of today. His tests are seen as mild today and it allows many "bad" generators to pass the tests.
2) Diehard: As computers become more advanced, more random numbers are consumed than ever before. Random numbers generators once were satisfactory are no longer good enough for sophisticated applications in physics, combinatorics and stochastic geometry. In 1995, Professor Marsaglia from Statistics Department in Florida State University introduced a suite of stringent tests that go beyond the approach Knuth's classical approaches and named it "Diehard Suites". They are stringent in the sense that they are difficult to pass. However, as Marsaglia retired and the suites have not been maintained in the past few years. Therefore it was not selected as the tests for this study.
3) National Institute of Standards and Technology (NIST): National Institute of Standards and Technology (NIST) is a measurement standard laboratory and it is a non-regulatory agency of the United States Department of Commerce. It promotes U.S. The institute's official mission is to "promote innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life". Released in 2001, the NIST Statistical Test Suites is a package consisting of 15 tests that were developed to test the randomness of arbitrary long binary

sequence produced by either hardware or software. The test suites is the result of collaboration between the Computer Security Division and the Statistical Engineering Division at NIST in response to a perceived need for a credible and comprehensive set of tests for binary (not uniform) random number generations. The test suites make use of both existing algorithm from past literatures and newly developed tests. NIST is now by and large the standard in the world of RNG testing. As a result, NIST suites are used as the comprehensive testing criteria to determine the quality of the RNG. The list of 15 tests in the suite is shown below and the purpose of each test is described briefly. Please refer to NIST Special Publication 800-22rev1a (dated April 2010) "A Statistical test Suite for the Validation of Random Number Generators and Pseudo random Number Generators for Cryptographic Applications" for detailed description for the purpose and procedure of each test.

1. **The Frequency (Monobit) Test**: Tests proportion of zeros and ones for the whole sequence
2. **Frequency Test within a block**: Tests the proportions of ones within M-bit Block
3. **The Run Test:** Test the total number of runs in the sequence, where a run is an uninterrupted sequence of identical bits
4. **Tests for the Longest-Run-of-Ones in a Block:** Test the longest run of ones within M-bit Block and consistency with theory
5. **The binary Matrix Rank Test**: Test rank of disjoint sub-matrices of the entire sequence and independence

6. **The Discrete Fourier Transform (Spectral) Test**: Tests the peak heights in the Discrete Fourier Transform of the sequence, to detect periodic features that indicates deviation of randomness
7. **The Non-overlapping Template Matching Test:** Tests the number of occurrences of a pre-specified target strings
8. **The overlapping Template Matching Test:** Test the number of occurrences of a pre-specified target strings. When window found, slide only one bit before the next search
9. **Maurer's "Universal Statistics" Test:** Tests the numbers of bits between matching patterns
10. **The Linear Complexity Test:** Tests the length of a linear feedback shift register, test complexity
11. **The Serial Test:** Test the frequency of all possible overlapping m-bit pattern
12. **The Approximate Entropy Test**: Tests the frequency of all possible overlapping m-bits pattern across the entire sequence
13. **The Cumulative Sums (Cusums) Test:** Tests maximal excursion from the random walk defined by the cumulative sum of adjusted (-1, +1) digits in the sequence
14. **The random Excursion Test:** Test the number of cycles having exactly K visits in a cumulative sum random walk
15. **The Random Excursions Variant Test:** Test the total number of times that a particular state is visited in a cumulative sum random walk

The software to test the binary random sequences are created and maintained by NIST engineers and it is published online free to users [19]. Software patch sts-2.1.1 publish in August 2010 is download and run in Linux machine as benchmark to test randomness. The result is reported as a pass/failure based on critical p-values and it is given by the software in *finalAnalysisReport.txt* file. Due to the limitation of space, the detailed description and observation from two of the tests are shown here.

1) Test #5: Random Binary Matrix Test

The binary Matrix Rank test checks linear dependence among fixed-length substrings of the original sequence. Matrices of successive zeros and ones are constructed and check the linear independence among the rows or columns. The deviation of the rank, or rank deficiency, of the matrices is calculated using a chi-square distribution. The matrices used in the test have dimensions of 32 by 32. The probabilities of matrices with the following ranks are shown below:

p_{32}=0.2888, p_{31}=0.5776, p_{30}=0.1284

To apply the Chi-Squared Test, use the classical statistics:

$$X^2 = \frac{(F_m - 0.2888N)^2}{0.2888N} + \frac{(F_{m-1} - 0.5776N)^2}{0.5776N} + \frac{(N - F_m - F_{m-1} - 0.1336N)^2}{0.1336N}$$

The p-value is calculated with the following expression:

$$p = \exp(-\frac{X^2(obs)}{2})$$

CASE 1: 10000 Binary Bits Generated from the Original Sequence

For this case, the random integers generated are converted to binary sequence on a one-to-one basis. A binary value of 0 is generated when the integer value is between 1 and 32

and a binary value of 1 is generated when the integer value is between 33 and 64. The results of the test is outputted by the software are shown below. The calculated p-value is less than 0.01, which indicates that the alternative hypothesis should be accepted rather than the null hypothesis.

```
COMPUTATIONAL INFORMATION:
       ----------------------------------------------
       (a) Probability P_32 = 0.288788
       (b)           P_31 = 0.577576
       (c)           P_30 = 0.133636
       (d) Frequency   F_32 = 0
       (e)           F_31 = 4
       (f)           F_30 = 5
       (g) # of matrices    = 9
       (h) Chi^2           = 12.27
       (i) NOTE: 784 BITS WERE DISCARDED.
       ----------------------------------------------
FAILURE       p_value = 0.002
```

The results indicate a pattern of linear independence with the tested sequence and show evidence of non-randomness. However, only 9 matrices were generated in this test and it is a relatively small sample. A large input data might be needed to make conclusive decisions and determine if it is a statistical anomaly.

CASE II: 60000 Binary Bits Generated from the Original Sequence

For this case, the integer generated on the interval between 1 and 64 are subtracted by 1 and then converted to a 6 bit binary output. As a result, a sequence with 60000 binary bits is generated from the algorithm. The results of the test is outputted by the software are shown on next page. The calculated p-value is less than 0.01, which indicates that the alternative hypothesis should be accepted rather than the null hypothesis.

COMPUTATIONAL INFORMATION:

(a) Probability P_32 = 0.288788
(b) P_31 = 0.577576
(c) P_30 = 0.133636
(d) Frequency F_32 = 4
(e) F_31 = 38
(f) F_30 = 16
(g) # of matrices = 58
(h) Chi^2 = 10.31
(i) NOTE: 608 BITS WERE DISCARDED.

FAILURE p_value = 0.0057

Although more bits were generated from the new algorithm, the Binary Rank Test still fails. It indicates that the mapping from random integer numbers to random binary sequence is not the cause of the failure. It shows that there exist three matrices with full rank compared to none in the first test. However, the deviation in rank compared to theoretical values is still significant that it causes the failure of the test with Chi-Square distribution. More statistical tests and further investigation is required to determine that if the sample size is the main cause of failure.

CASE III: A Million Bits Generated from the New Experiment

In order to evaluate whether dependence exists in the sequence, more experimental data are generated using the same algorithm with a new bit. One million random numbers between the range of 1 to 64 are generated and one to one conversion is used in generating the binary sequence. The results of the test is outputted by the software are shown on next page. There are in total 976 matrices with dimension 32 by 32 being tested. The deviations of ranks are much smaller compared to the two sequences from the original data. The p-value of this test is 0.567, which is well above the critical value of 0.01. This indicates that the sequence is considered random and there is no evidence

COMPUTATIONAL INFORMATION:

(a) Probability P_32 = 0.288788
(b) P_31 = 0.577576
(c) P_30 = 0.133636
(d) Frequency F_32 = 296
(e) F_31 = 556
(f) F_30 = 124
(g) # of matrices = 976
(h) Chi^2 = 1.132
(i) NOTE: 576 BITS WERE DISCARDED.

SUCCESS p_value = 0.567775

showing dependence with previous observations. This test shows that previous failures are mainly due to the length of the data or the nature of that specific sequence. The RTN based random number generating algorithm is acceptable and passes the binary rank test.

2) Test #12: Approximate Entropy Test

The Approximate Entropy Test focuses on the frequency of all possible m-bit patterns across the entire sequence. The purpose of the test is to compare the frequency of overlapping blocks of two consecutive/adjacent lengths (m and m+1, note m=4 for this specific test) against the expected random sequence. The count of all possible combination of bit lengths m and m+1 is tabulated then the entropy for length m and m+1 is calculated accordingly. The Test Statistics is given by:

$X^2 = 2n[\log 2 - \mathrm{ApEn}(m)]$ where ApEn(m) = entropy(m) - entropy(m+1)

Then the P-value is computed as following: P-value = igamc(2^{m-1}, $X^2/2$)

where igmac denotes incomplete gamma distribution.

The result outputted from the software is shown on next page:

```
 APPROXIMATE ENTROPY TEST
       ---------------------------------------------
       COMPUTATIONAL INFORMATION:
       ---------------------------------------------
       (a) m (block length)    = 4
       (b) n (sequence length) = 10000
       (c) Chi^2               = 16.703866
       (d) Phi(m)              = -2.771524
       (e) Phi(m+1)            = -3.463836
       (f) ApEn                = 0.692312
       (g) Log(2)              = 0.693147
       ---------------------------------------------
SUCCESS       p_value = 0.405005
```

The p-value is greater than 0.01, which indicates that the null hypothesis is accepted. Therefore, the given sequence passed the Approximate Entropy Test and shows no evidence s of non-randomness.

Odd-Even Based Sampling:

In addition to the random number generating scheme using the transformation from exponential to uniform distribution, an extremely simple odd-even based sampling is performed to test the randomness of the sequence. Similar to the previous algorithm, the times that the error patterns stay in up/down states are tabulated as an integer multiple of sampling times. The binary sequences are generated based on whether this integer number is even or odd. Conceptually, this set of binary sequence is random due to the fact that Random Telegraph Noise is random. With this sampling approach, the original sequences with length 10000 and the new sequence with lengths 1 million are both tested with the comprehensive NIST suites. Both sequences pass all 15 tests in the suites and it confirms with the early assumption. The advantage of this sampling algorithm is obviously its simplicity thus it leads to a faster number generation. However, the

throughput produced from this sampling scheme is much smaller compared to the algorithm presented earlier, where 6 bits could be generated with each time count.

Summary:

The original sequence passed 14 out of 15 tests in the NIST binary sequence testing suites. It failed Test #5 Random Binary Matrix Test as shown above. The failed test focuses on linear dependence of sub-block within the sequence. After further investigation, only nine matrices are generated from the small sample size and it might not fully represent the statistical distribution of matrices ranks. With the new experimental data about one million bits, the new sequence passes the binary rank test with a p-value 0.57. Therefore, it shows that the random number generator presented indeed performs well and the results are truly random. However, the testing is still at beginning stage and more experimental data from many bits are needed to test the consistence of the results. More data sample should also be collected and the test should be conducted again to make sure the result is valid. Furthermore, in the event that some tests that fail, additional numerical experiments should be conducted on different samples of the RNG to determine whether the phenomenon was a statistical anomaly or indeed it is a clear evidence of non-randomness.

4.6 Comparison with other RNGs

With the proposed RNG using RTN, it is essential to compare it with other state of art RNG's. In this study, the test results are compared with ones from the MATLAB default random number generator and Random.org generator. It is worthwhile to note that

MATLAB uses a pseudorandom generator and Random.org is a true RNG. The descriptions of generating algorithms are stated below.

1) MATLAB default RNG

   MATLAB default RNG is a pseudorandom number generator and it uses an algorithm called "Mersenne Twister" with seed 0. The algorithm was developed in 1997 by two Japanese mathematicians. It is based on a matrix linear recurrence over a finite binary field F_2. It provides for fast generation of very high-quality pseudorandom numbers as it was designed specifically to rectify flows in older algorithms.

2) Random.org RNG

   Random.org's random numbers are generated using a TRNG. Its source of entropy is atmospheric noise. It is obtained by tuning a radio to a station that no one is using. It is then played into a workstation server where a program converts it to an 8-bit mono-signal at a frequency of 8 kHz. Then the first seven bits are

Table 11: Comparison of RNG using Theoretical Statistics Test

	Test Statistics			
Test	Critical value	RTN	MATLAB	Random.org
Chi-Squared Test	$x^2_{0.05,9} = 16.92$	$x^2_{0.05,9}$=15.13	$x^2_{0.05,9}$= 8.21	$x^2_{0.05,9} = 7.04$
		Accept H_0	Accept H_0	Accept H_0
Test of runs above and below the Median	-1.96<z <1.96	z = 0.552	z = 0.365	z = -0.351
		Accept H_0	Accept H_0	Accept H_0
The Overlapping Sums Test	-1.96<z <1.96	z = 0.937	z = -0.682	z = -0.452
		Accept H_0	Accept H_0	Accept H_0

Table 12: Comparison of RNG using NIST Test Suites

Test #	Test Name	RTN	MATLAB	Random.org
1	Frequency test	✓	✓	✓
2	Test for Frequency within a block	✓	✓	✓
3	Runs Test	✓	✓	✓
4	Test for Longest Run of Ones in a block	✓	✓	✓
5	Random Binary Matrix Rank Test	✓	✓	✓
6	Discrete Fourier Transform Test	✓	✓	✓
7	Non-Overlapping Template Test	✓	✓	✓
8	Overlapping Template Test	✓	✓	✓
9	Maurer's Universal Statistics Test	✓	✓	✓
10	Linear Complexity Test	✓	✓	✓
11	Serial Test	✓	✓	✓
12	Approximate Entropy	✓	✓	✓
13	Cumulative Sum Test	✓	✓	✓
14	Random Excursion Test	✓	✓	✓
15	Random Excursions Variant Test	✓	✓	✓

discarded and the remaining bits are collected. This stream of bits has very high entropy. The last step is to perform a skew correction on the stream to ensure that an even distribution of 0's and 1's.

Sequences of 10000 random numbers are generated using each of the three schemes of RNGs. The three theoretical tests are tested on the samples and the comparisons of the three theoretical statistics tests results are shown in **Table 11** on previous page. The NIST suites to test binary random sequences are also used and the comparisons of results are shown in **Table 12** on previous page as well.

Based on the results shown in **Table 11** above, it is observed that all three RNGs passed the theoretical statistics tests with 5% significant level. With the NIST random sequence test, it is shown that the original sequence generated by RTN based RNG passed 14 out of 15 tests but the new sequence with large sample size passes the failing tests. The pseudorandom number generator and the random.org's TRNG passed all 15 tests. It is difficult to make a definitive conclusion based on the small test sample. The preliminary result indicates the trend that the RNG based on RTN satisfy the needs of random number generation. However, the speed of number generations for RTN based RNG is significantly slower compared to MATLAB's pseudorandom algorithm as well as the random.org scheme. The methodology applied in this chapter is still primitive in nature. The process in selecting specific bits to generate bit error pattern, the signal processing and analyzing algorithms, the measurements methodology implied in this original study still requires improvement. A skew correction mechanism should be investigated and see if it could help the randomness of the sequence. In addition, more detailed testing with a large quantity of bits within different blocks should be tested as well as different flash memory chips from various vendors. The study demonstrates the existence of RTN noise in Flash memory chips and it shows the encouraging and promising trends of generating random numbers based on this physical phenomenal.

5. Conclusion and Future Work

5.1 Conclusion

In this study, an innovative model to characterize the behavior of flash memory cells in stress is proposed to study the reliability issue. The simplified theoretical model includes three main factors: stress induced leakage current, random telegraph noises and peripheral circuits caused reading errors. Experimental data of error patterns after ten million erase cycles from one physical flash memory chip is collected and analyzed. Simulated Annealing algorithm is implemented to extract the parameters in the model. MATLAB simulations are then performed in order to validate the model. The proposed model fits well with the average trial experiment but shows discrepancy in individual trials as erratic erase and various initial threshold voltages might contribute to the unpredictability. It is concluded that the proposed model gives a basic and accurate representation of the major contributing factors of flash cell failures, but it is inconclusive to say that this model fully represent the failure mechanism of every single cells.

Random Number Generations using the Random Telegraph Noises phenomenal is an interesting application based on the proposed model. A novel approach of partial erase with relatively small stress is implemented in order to extract the best one-trap behaviors in error patterns suitable for random number generator. The time threshold spend in up or down states are measured and modeled as an exponential distribution. The random variable is then transformed into a uniform distribution in the interval between 0 and 1. The sequence of random numbers is then tested using a rigorous approach. Exploratory Data Analysis, Theoretical Statistical Test and industry standard National Institute of

Standard and Technology random binary sequence testing suites for cryptography applications are all used as means to verify the quality of the random numbers. The testing work is still at preliminary stage where only a limited amount of number is tested. It is shown the feasibility of random number generation using the RTN phenomenal as it satisfies the main requirements of randomness test, but it fails 2 out of 15 tests in a more sophisticated random testing scheme. The reason of failure is still unknown and more numerical tests should be performed to see if it is a statistical anomaly or indeed the sequence shows evidence of non-randomness.

5.2 Future Work

Although the study shown in this thesis demonstrated an enormous accomplishment, there are still many future researches ahead in order to have a comprehensive understanding on the reliability issue. The proposed model is unable to explain the weird and inconsistent behavior of stress induced leakage current after each erase operation. It is speculated that transient current might play a significant during the short period after erase operation and it should be taken into consideration as part of the model. This is only one of the possible improvements in the model that future project should work on in the near future. It is still an unknown question whether this model comprehensively characterize the behavior of flash cells in stress. More experiments should be performed with various manufacturing technology in order to show conclusive results.

There are many applications with the proposed reliability model. Random Number Generation is simply one of them that is in focus for this study. The initial study

shows promising results on the feasibility of RNG based on RTN, but more systematic testing should be performed as well with the existing infrastructures. In addition, skew correction algorithm should be investigated and tested to see if it helps improve the performance of the RNGs.

Digital signature within a flash chip is another mind-boggling application that future research could work on. The numerical values of parameters for various bits in different blocks could be used as a unique signature of each flash chip and information could be hidden within the physical layer of flash chips as well. The digital signature application could also be as a way to validate and different authentic flash chips against fake ones. In addition, an error correction code to improve the mean time to failure of flash memory chips is another challenging project in industry. There are enormous possibilities ahead with the reliability issues in flash memory cells as the dimension of device size gets reduce further. The researches on this topic will continue and it will impose unique but challenging problems to scientists and engineers worldwide.

Appendix I: MATLAB simulation of the Proposed Model

```
len=length(t);

 for i=1:len
        Vth(i)=Vth0*exp(-t(i)*tao1);
end

tdis=zeros(1,len);

for ss=1:200
up=exprnd(up_mean, [1, len]);
down=exprnd(down_mean, [1,len]);

for i=1:len
       up(i)=round(up(i));
       down(i)=round(down(i));
end
RTN=zeros(1,len);
 i=1;
 ptr=1;  %pointer
 while(i<len)
        j=down(ptr);
        i=i+j;
        j=up(ptr)+i;
        while(i<j)
                RTN(i)=deltaV;
                i=i+1;
        end
        ptr=ptr+1;
 end
 vtemp=Vth+RTN(1:len);

 for i=1:len
         ferr(i)=1-1/(exp(alpha*(vtemp(i)-Vread))+1);
         temp=rand(1,1);
         if temp<=ferr(i)
                 cell1(i)=1;
         tdis(i)=tdis(i)+1;
         end
 end
```

Appendix II: Exhaustive Search Algorithm

```
for i=1:length(alpha)
    for r=1:length(ratio_enum)
        ratio=ratio_enum(r);
        upmean=downmean*(1-ratio)/ratio;
    for j=1:length(downmean)
        for k=1:length(Vth0)
            for m=1:length(tao1)
                for p=1:length(Vread)
                    err_t=0;
                    for z=1:length(input)
                        temp1=(1-1/(exp((alpha(i))*(Vth0(k)*exp(-1*z*dt/tao1(m))-
Vread(p)))+1));
                        temp2=(1-1/(exp((alpha(i))*(Vth0(k)*exp(-z*dt/tao1(m))-
Vread(p)+deltaV))+1)); %%
                        P1=ratio+(1-ratio)*exp(-z*dt*(1/downmean(j)+1/upmean(j)));
                        error=temp1*P1+temp2*(1-P1);
                        error=(input(z)-error)^2
                        err_t=error+err_t;
                        loop=loop+1;
                    end
                    if (count==0)
                        min_e=err_t;
                        count=count+1;
                    else
                        if(err_t<min_e)
                            min_e=err_t;
                            ind_i=i;
                            ind_j=j;
                            ind_k=k;
                            ind_m=m;
                            ind_p=p;
                            ind_ratio=r;
                        end
                    end
                end
            end
        end
    end
end
```

Appendix III: Simulated Annealing Algorithm

```
while (count<count_max && err_cur > err_min)

    if (mod(count,10000)==0)
        fprintf('count=%d, Error=%.4d\n',count,err_cur);
    end

    num=randi(12,1,1);

    if (num==1 && cur_i~=length(Vread))
        cur_i=cur_i+1;
    elseif (num==2 && cur_i~=1)
        cur_i=cur_i-1;
    elseif (num==3 && cur_j~=length(Vth0))
        cur_j=cur_j+1;
    elseif (num==4 && cur_j~=1)
        cur_j=cur_j-1;
    elseif (num==5 && cur_k~=length(tao1))
        cur_k=cur_k+1;
    elseif (num==6 && cur_k~=1)
        cur_k=cur_k-1;
    elseif (num==7 && cur_m~=length(alpha))
        cur_m=cur_m+1;
    elseif (num==8 && cur_m~=1)
        cur_m=cur_m-1;
    elseif (num==9 && cur_p~=length(ratio_enum))
        cur_p=cur_p+1;
    elseif (num==10 && cur_p~=1)
        cur_p=cur_p-1;
    elseif (num==11 && cur_q~=length(downmean))
        cur_q=cur_q+1;
    elseif (num==12&& cur_q~=1)
        cur_q=cur_q-1;
    end

    ratio=ratio_enum(cur_p);
    upmean=downmean(cur_q)*(1-ratio)/ratio;
    err_t=0;

    for z=1:length(input)
        temp1=(1-1/(exp((alpha(cur_m))*(Vth0(cur_j)*exp(-z*dt*tao1(cur_k))-
Vread(cur_i)))+1));
        temp2=(1-1/(exp((alpha(cur_m))*(Vth0(cur_j)*exp(-z*dt*tao1(cur_k))-
Vread(cur_i)+deltaV))+1));
```

```
        P1=ratio+(1-ratio)*exp(-z*dt*(1/downmean(cur_q)+1/upmean));
        error=temp1*P1+temp2*(1-P1);
        error=(input(z)-error)^2;
        err_t=error+err_t;
    end

    if (err_t>err_cur)
        acc_prob=exp((err_cur-err_t)/T_const);
        if (acc_prob<rand(1,1))
            if (num==1 && cur_i~=length(Vread))
                cur_i=cur_i-1;
            elseif (num==2 && cur_i~=1)
                cur_i=cur_i+1;
            elseif (num==3 && cur_j~=length(Vth0))
                cur_j=cur_j-1;
            elseif (num==4 && cur_j~=1)
                cur_j=cur_j+1;
            elseif (num==5 && cur_k~=length(tao1))
                cur_k=cur_k-1;
            elseif (num==6 && cur_k~=1)
                cur_k=cur_k+1;
            elseif (num==7 && cur_m~=length(alpha))
                cur_m=cur_m-1;
            elseif (num==8 && cur_m~=1)
                cur_m=cur_m+1;
            elseif (num==9 && cur_p~=length(ratio_enum))
                cur_p=cur_p-1;
            elseif (num==10 && cur_p~=1)
                cur_p=cur_p+1;
            elseif (num==11 && cur_q~=length(downmean))
                cur_q=cur_q-1;
            elseif (num==12&& cur_q~=1)
                cur_q=cur_q+1;
            end
        else
            err_cur=err_t;
        end
    else
        err_cur=err_t;
    end
    count=count+1;
end
```

BIBLIOGRAPHY

1. Nathan Myhrvold, "Moore's Law Corollary: Pixel Power". *New York Times*, June 7, 2006

2. D.A. Baglee and M. Smayling, "The Effects of Write/Erase Cycling on Data Loss in EEPROMs," *IEEE IEDM Tech. Dig.,* pp. 624–626, 1985.

3. S. Aritome, R. Shirota, G. Hemink, T. Endoh, and F. Masuoka, "Reliability Issues of Flash Memory Cells," *Proc. IEEE,* Vol. 81, No. 5, pp. 776–787, May 1993.

4. J. Kim, J. Choi, W. Shin, D. Kim, H. Kim, K. Mang, S. Ahn, and O. Kwon, "Scaling Down of Tunnel Oxynitride in NAND Flash Memory: Oxynitride Selection and Reliabilities," *Proc. of International Reliability Physics Symp.,* p. 12, 1997.

5. Hoefler, J.Higman, T.Harp, and P.Kuhn, "Statistical Modeling of the Program/Erase Cycling Acceleration of Low Temperature Data Retention in Floating-Gate Nonvolatile Memories," *Proc. International Reliability Physics Symp.,* pp. 21–25, 2002.

6. H.Belgal, N.Righos, I.Kalstirsky, J.Peterson, R.Shiner, N.Mielke, "A New Reliability Model for Post-Cycling Charge Retention of Flash Memories," *Proc. IEEE International Reliability Physics Symp.,* pp. 7–20, 2002.

7. D.Ielmini, A.Ghetti, A.Spinelli, and A.Visconti, "A Study of Hot-Hole Injection during Programming Drain Disturb in Flash Memories,"*IEEE Trans. Elect. Dev.,* Vol. 53, No. 4, pp. 668–676, April 2006.

8. M.Suhail, T.Harp, J.Bridwell, and P.J.Kuhn, "Effects of Fowler Nordheim Tunneling Stress vs.Channel Hot Electron Stress on Data Retention Characteristics of Floating Gate Non-Volatile Memories," *Proc. International Reliability Physics Symp.,* pp. 439–440, 2002

9. R.Degraeve, F.Schuler, B.Kaczer, M.Lorenzini, D.Wellekens, P.Hendricks, M.van Duuren, G.J.M.Dormans, J.Van Houdt, L.Haspeslagh, G.Groeseneken, and G.Tempel, "Analytical Percolation Model for Predicting Anomalous Charge Loss in Flash Memories," *IEEE Trans. Elect. Dev.,* Vol. 51, No. 9, pp. 1392–1400, Sept. 2004.

10. D.Ielmini, A.Spinelli, A.Lacaita, and M.van Duuren, "Impact of Correlated Generation of Oxide Defects on SILC and Breakdown Distributions," *IEEE Trans. Elect Dev.,* Vol. 51, No. 8, pp. 1281–1287, Aug. 2004.

11. C.H.Lee, J.Choi, C.Kang, Y.Shin, J.S.Lee, J.Sel, J.Sim, S.Jeon, B.I.Choe, D.Bael, K.Park, and K.Kim, "Multi-Level NAND Flash Memory with 63nm-Node TANOS (SiOxide-SiN-Al_2O3-TaN) Cell Structure," *IEEE Symposium on VLSI Technology*, pp. 24–25, 2006.

12. H.Kurata, K.Otsuga, A.Kotabe, S.Kajiyama, T.Osabe, Y.Sasago, S.Narumi, K.Tokami, S.Kamohara, and O.Tsuchiya, "The Impact of Random Telegraph Signals on the Scaling of Multilevel Flash Memories," *IEEE Symposium on VLSI Circuits*, pp. 112–113, 2006.

13. H.Kurata, K.Otsuga, A.Kotabe, S.Kajiyama, T.Osabe, Y.Sasago, S.Narumi, K.Tokami, S.Kamohara, and O.Tsuchiya, "The Impact of Random Telegraph Signals on the Scaling of Multilevel Flash Memories," *IEEE Symposium on VLSI Circuits*, pp. 112–113, 2006.

14. Kanter, Ido; Aviad, Yaara; Reidler, Igor; Cohen, Elad; Rosenbluh, Michael, *"An optical ultrafast random bit generator"*, Nature Photonics, Volume 4, Issue 1, pp. 58-61 (2010).

15. A. Das and B. K. Chakrabarti (Eds.), *Quantum Annealing and Related Optimization Methods*, Lecture Note in Physics, Vol. 679, Springer, Heidelberg (2005)

16. National Institute of Standard and Technology, *A Statistic Test Suits for Random and Pseudorandom Number Generators for Cryptographic Applications*, 2001

17. Vattulainen et al., *A Comprehensive Study of Some Pseudorandom Generators*, Department of Electrical Engineering, Helsinki University of Technology, August 1993

18. Knuth, Donald E., *The Art of Computer Programming - Seminumerical Algorithm.* Vol 2 Chapter 3 *Random Numbers* pg1-184. 1997.

19. National Institute of Standard and Technology, *A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications*, April 2010 , "http://csrc.nist.gov/groups/ST/toolkit/rng/documents/SP800-22rev1a.pdf"